Symbolic verification of communicating systems with probabilistic message losses: liveness and fairness*

C. Baier[1], N. Bertrand[2], and Ph. Schnoebelen[2]

[1] Universität Bonn, Institut für Informatik I, Germany
[2] LSV, ENS de Cachan & CNRS, France

Abstract. NPLCS's are a new model for nondeterministic channel systems where unreliable communication is modeled by probabilistic message losses. We show that, for ω-regular linear-time properties and finite-memory schedulers, qualitative model-checking is decidable. The techniques extend smoothly to questions where fairness restrictions are imposed on the schedulers. The symbolic procedure underlying our decidability proofs has been implemented and used to study a simple protocol handling two-way transfers in an unreliable setting.

1 Introduction

Channel systems [15] are systems of finite-state components that communicate via asynchronous unbounded fifo channels. *Lossy* channel systems [17, 6], shortly LCS's, are a special class of channel systems where messages can be lost while they are in transit. They are a natural model for fault-tolerant protocols where communication is not supposed to be reliable (see example in Fig. 1 below). Additionally, the lossiness assumption makes termination and safety properties decidable [22, 17, 6, 4, 20, 8] while reliable, i.e., non-lossy, systems are Turing-powerful.

LCS's are a convenient model for verifying safety properties of asynchronous protocols, and this can be automated [4]. However, they are not adequate for verifying liveness and progress properties: firstly these properties are undecidable for LCS's [5], and secondly the model itself is too pessimistic when liveness is considered. Indeed, to ensure any kind of progress, one must assume that at least some messages will not be lost. This is classically obtained via fairness assumptions on message losses [18] but fairness assumptions in LCS's make decidability even more elusive [5, 21].

Probabilistic LCS's, shortly PLCS's, are LCS's where message losses are seen as faults having a *probabilistic* behavior [27, 10, 31, 1, 29, 2, 7]. Thanks to its probabilistic framework, this model automatically fulfills strong fairness conditions on the message losses. Additionally it allows one to state so-called *qualitative* questions, whether a linear-time property will be satisfied "*with probability 1*", that are decidable. However, PLCS's are not a realistic model for protocols because they consider that the choices between different actions are made probabilistically rather than nondeterministically. When modeling communication protocols, *nondeterminism* is an essential feature. It

* The first author is supported by the DFG-NWO project VOSS II and the DFG-project PROBPOR. The last two authors were supported by the ACI Sécurité & Informatique project Persée.

is used to model the interleaved behavior of distributed components, to model an unknown environment, to delay implementation choices at early stages of the design, and to abstract away from complex control structures at later stages.

This prompted us to introduce NPLCS's, i.e., channel systems where message losses are probabilistic and actions are nondeterministic [13, 14]. These systems give rise to infinite-state Markov decision processes, and are a more faithful model for analyzing protocols. The drawback is that they raise very difficult verification problems.

Qualitative verification for NPLCS's. Our early results in [14] rely on the assumption that idling was always a possible choice. This simplifies the analysis considerably, but is an overkill: a necessary ingredient for most liveness properties of a compound system is the inherent liveness of the components, which disappears if they can idle.

We developed new techniques and removed the idling limitation in [9] where we show that decidability can be maintained if we restrict our attention to *finite-memory* schedulers (strategies for the nondeterministic choices). This seems like a mild restriction, and we adopt it in this paper since we aim for automatic verification.

Our contributions. In this paper we extend the preliminary work from [9] in three directions: (1) We allow linear-time formulas referring to the contents of the channels rather than just the control locations. We did not consider this extension earlier because we lacked the techniques for proving the convergence of fixpoint computations. However, the extension is required in practical applications where fairness properties have to express that "a rule is firable," which depends on channel contents for read actions. (2) We develop symbolic representations and algorithms for sets of NPLCS configurations. These algorithms have been implemented in a prototype tool that we use to analyze a simple communication protocol. (3) We consider qualitative verification with quantification over *fair* schedulers, i.e., schedulers that generate fair runs almost surely.

Outline of the paper. Section 2 recalls the necessary technical background for nondeterministic probabilistic channel systems, and section 3 introduces the new symbolic framework we use for handling sets of configurations. We present our decidability results in sections 4 (for finite-memory schedulers) and 5 (for fair schedulers). Finally we apply our algorithms on Pachl's protocol in section 6. All proofs omitted in this extended abstract can be found in the complete version available on the web.

2 Nondeterministic probabilistic channel systems

We assume the reader has some familiarity with the verification of Markov decision processes, or MDPs, (otherwise see [11]) and refer to [9] for complete definitions regarding our framework. Here we recall the main definitions and notations without motivating or illustrating all of them.

Lossy channel systems. A lossy channel system (a LCS) is a tuple $\mathcal{L} = (Q, \mathsf{C}, \mathsf{M}, \Delta)$ of a finite set $Q = \{p, q, \ldots\}$ of control *locations*, a finite set $\mathsf{C} = \{c, \ldots\}$ of *channels*, a finite *message alphabet* $\mathsf{M} = \{m, \ldots\}$ and a finite set $\Delta = \{\delta, \ldots\}$ of *transition rules*. Each rule has the form $q \xrightarrow{op} p$ where op is an *operation* of the form $c!m$ (sending message m along channel c), $c?m$ (receiving message m from channel c), or $\surd$ (an internal action with no communication). For example, the protocol displayed in Fig 1, is naturally modeled as a LCS: building the asynchronous product of the two processes $P_{\mathtt{L}}$ and $P_{\mathtt{R}}$ yields a bona fide LCS with two channels and a five-message alphabet $\mathsf{M} = \{\mathtt{a}_0, \mathtt{a}_1, \mathtt{d}_0, \mathtt{d}_1, \mathtt{eod}\}$.

Operational semantics. A *configuration* of $\mathcal{L}$ as above is a pair $s = (q, w)$ of a location and a channel valuation $w : \mathsf{C} \to \mathsf{M}^*$ associating with any channel its current content (a sequence of messages). $\mathsf{M}^{*\mathsf{C}}$, or M^* when $|\mathsf{C}| = 1$, denotes the set of all channel valuations, and Conf the set of all configurations. ε denotes both the empty word and the empty channel valuation. The size $|s|$ of a configuration is the total number of messages in s. The rules of $\mathcal{L}$ give rise to transitions between configurations in the obvious way [9]. We write $\Delta(s)$ for the set of rules $\delta \in \Delta$ that are enabled in configuration s.

We write $s \xrightarrow{\delta}_{\text{perf}} s'$ when s' is obtained by firing δ in s. The "perf" subscript stresses the fact that the step is perfect, i.e., no messages are lost. However, in lossy systems, arbitrary messages can be lost. This is formalized with the help of the subword ordering: we write $\mu \sqsubseteq \mu'$ when μ is a subword of μ', i.e., μ can be obtained by removing (any number of) messages from μ', and we extend this to configurations, writing $(q, w) \sqsubseteq (q', w')$ when $q = q'$ and $w(c) \sqsubseteq w'(c)$ for all $c \in \mathsf{C}$. As a consequence of Higman's Lemma, $\sqsubseteq$ is a well-quasi-order (a *wqo*) between configurations of $\mathcal{L}$. Now, we define *lossy steps* by letting $s \xrightarrow{\delta} s'' \overset{\text{def}}{\Leftrightarrow}$ there is a perfect step $s \xrightarrow{\delta}_{\text{perf}} s'$ such that $s'' \sqsubseteq s'$. This gives rise to a labeled transition system $LTS_{\mathcal{L}} \overset{\text{def}}{=} (\mathsf{Conf}, \Delta, \to)$. Here the set Δ of transition rules serves as action alphabet. In the following we assume that for any location $q \in Q$, Δ contains at least one rule $q \xrightarrow{op} p$ where op is not a receive operation. This hypothesis ensures that $LTS_{\mathcal{L}}$ has no deadlock configuration and makes the theory smoother. It is no real loss of generality as demonstrated in [2, § 8.3].

An example. Pachl's protocol [22] handles two-way communications over lossy channels and is our case study for our algorithms. It consists of two identical processes, $P_{\mathtt{L(eft)}}$ and $P_{\mathtt{R(ight)}}$, that exchange data over lossy channels using an acknowledgment mechanism based on the alternating bit protocol. See Fig 1 below. The actual contents of the data messages is abstracted away, and we just use $\mathtt{d}_0, \mathtt{d}_1 \in \mathsf{M}$ to record the alternating control bit. Messages $\mathtt{a}_0, \mathtt{a}_1 \in \mathsf{M}$ are the corresponding acknowledgments. The protocol starts in configuration $(\mathtt{L0}, \mathtt{R4})$ where $P_{\mathtt{L}}$ is the sender and $P_{\mathtt{R}}$ the receiver. At any time (provided its last data message has been acknowledged) the sender may signal the end of its data sequence with the $\mathtt{eod} \in \mathsf{M}$ control message and then the two processes swap their sending and receiving roles. Note that $\mathtt{eod}$ does not need to carry a control bit, and that its correct reception is not acknowledged. In section 6 we explain how such a two-process protocol is modeled as an LCS, and give some outcomes of our automated analysis.

From LCS's to NPLCS's. A NPLCS $\mathcal{N} = (\mathcal{L}, \tau)$ is a LCS $\mathcal{L}$ further equipped with a *fault rate* $\tau \in (0,1)$ that specifies the probability that a given message stored in one of the message queues is lost during a step [13, 14]. The operational semantics of NPLCS's

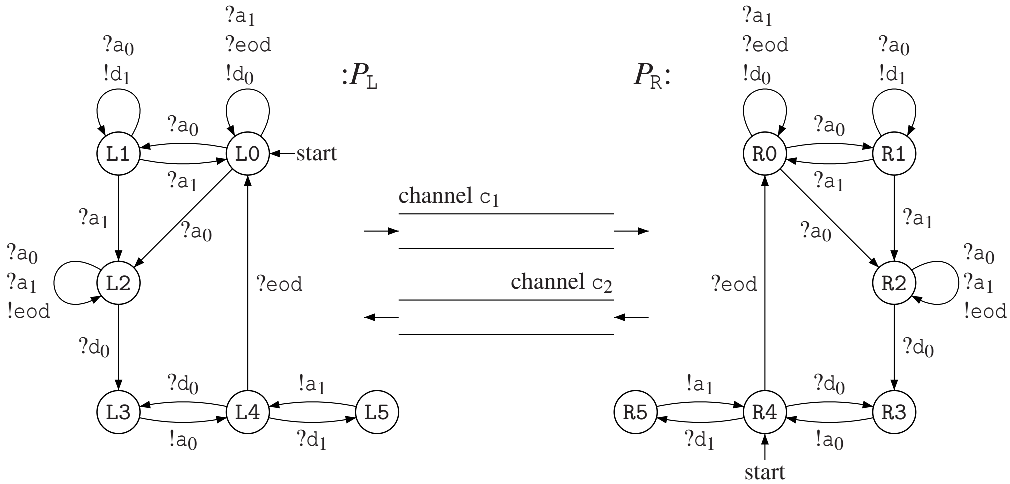

Fig. 1. Pachl's communication protocol, from [22].

has the form of an infinite-state Markov decision process $MDP_{\mathcal{N}} \stackrel{\text{def}}{=} (\mathsf{Conf}, \Delta, \mathbf{P}_{\mathcal{N}})$. The stepwise probabilistic behavior is formalized by a three-dimensional transition probability matrix $\mathbf{P}_{\mathcal{N}} : \mathsf{Conf} \times \Delta \times \mathsf{Conf} \to [0,1]$. For a given configuration s and an enabled rule $\delta \in \Delta(s)$, $\mathbf{P}_{\mathcal{N}}(s,\delta,\cdot)$ is a distribution over Conf, while $\mathbf{P}_{\mathcal{N}}(s,\delta,\cdot) = 0$ for any transition rule δ that is not enabled in s. The intuitive meaning of $\mathbf{P}_{\mathcal{N}}(s,\delta,t) = \lambda > 0$ is that with probability λ, the system moves from configuration s to configuration t when δ is the chosen transition rule in s.

For lack of space, this extended abstract omits the technically heavy but quite natural definition of $\mathbf{P}_{\mathcal{N}}$, and only lists its two essential properties:
1. the labeled transition system underlying $MDP_{(\mathcal{L},\tau)}$ is exactly $LTS_{\mathcal{L}}$.
2. the set $Q_\varepsilon = \{(q,\varepsilon) \mid q \in Q\}$ of configurations where the channels are empty is an *attractor*, i.e., from any starting configuration, Q_ε will eventually be visited with probability 1 [2, 7].

Schedulers and probability measure. The nondeterminism in an MDP is resolved by a *scheduler*, also often called "adversary", "policy" or "strategy". Here a "scheduler" is a *history-dependent deterministic scheduler* in the classification of [28]. Formally, a scheduler for $\mathcal{N}$ is a mapping $\mathcal{U}$ that assigns to any finite path π in $\mathcal{N}$ a transition rule $\delta \in \Delta$ that is enabled in the last state of π. The given path π specifies the history of the system, and $\mathcal{U}(\pi)$ is the rule that $\mathcal{U}$ chooses to fire next. A scheduler $\mathcal{U}$ only gives rise to certain paths: we say $\pi = s_0 \xrightarrow{\delta_1} s_1 \xrightarrow{\delta_2} \cdots$ is *compatible with* $\mathcal{U}$ or, shortly, is a $\mathcal{U}$*-path*, if $\mathbf{P}_{\mathcal{N}}(s_{i-1},\delta_i,s_i) > 0$ for all $i \geq 1$, where $\delta_{i+1} = \mathcal{U}(s_0 \xrightarrow{\delta_1} \cdots \xrightarrow{\delta_i} s_i)$ is the rule chosen by $\mathcal{U}$ at step i along π. In practice, it is only relevant to define how $\mathcal{U}$ evaluates on finite $\mathcal{U}$-paths.

A *finite-memory*, or fm-, scheduler $\mathcal{U} = (U, D, \eta, u_0)$ is specified via a finite set U of *modes*, a *starting mode* $u_0 \in U$, a *decision rule* $D : U \times \mathsf{Conf} \to \Delta$ choosing the next rule $D(u,s) \in \Delta(s)$ based on the current mode and the current configuration, and a *next-mode function* $\eta : U \times \mathsf{Conf} \to U$ specifying the mode-changes of $\mathcal{U}$. The modes are used to store some relevant information about the history. An fm-scheduler $\mathcal{U}$ is *memoryless* if it has a single mode: then $\mathcal{U}$ is not history-dependent and can be specified more simply as a mapping $\mathcal{U} : \mathsf{Conf} \to \Delta$.

Now, given an NPLCS $\mathcal{N}$, a starting configuration $s = s_0$ and a scheduler $\mathcal{U}$, the behavior of $\mathcal{N}$ under $\mathcal{U}$ can be formalized by an infinite-state Markov chain $MC_{\mathcal{U}}$. For arbitrary schedulers, the states of $MC_{\mathcal{U}}$ are finite paths in $\mathcal{N}$, while for fm-schedulers it is possible to consider pairs (u,s) of a mode of $\mathcal{U}$ and a configuration of $\mathcal{N}$. One may now apply the standard machinery for Markov chains and define (for fixed starting configuration s) a sigma-field on the set of infinite paths starting in s and a probability measure on it, see, e.g., [28, 23, 11]. We shall write $\Pr_{\mathcal{U}}\big(s \models \ldots\big)$ to denote the standard probability measure in $MC_{\mathcal{U}}$ with starting state s.

LTL/CTL-notation. We use simple LTL and CTL formulas to denote properties of respectively paths and configurations in MDP_L. Here configurations and locations serve as atomic propositions: for example $\Box\Diamond s$ (resp. $\Box\Diamond q$) means that $s \in \mathsf{Conf}$ (resp. $q \in Q$) is visited infinitely many times, and q Until s means that the control location remains q until configuration s is eventually reached. These notations extend to sets and, for $T \subseteq \mathsf{Conf}$ and $P \subseteq Q$, $\Box\Diamond T$ and $\Box\Diamond P$ have the obvious meaning. For $P \subseteq Q$, P_ε is the set $\{(p,\varepsilon) \mid p \in P\}$ so that $\Diamond Q_\varepsilon$ means that eventually a configuration with empty channels is reached. It is well-known that for any scheduler $\mathcal{U}$, the set of paths starting in some configuration s and satisfying an LTL formula, or an ω-regular property, φ is measurable [32, 16]. We write $\Pr_{\mathcal{U}}\big(s \models \varphi\big)$ for this measure.

Reachability analysis. For a set $A \subseteq \mathsf{Conf}$ and a rule $\delta \in \Delta$, we let $Pre[\delta](A) \stackrel{\text{def}}{=} \{s \mid \exists t \in A, s \xrightarrow{\delta} t\}$ denote the set of configurations from which A can be reached in one step with rule δ. $Pre(A) \stackrel{\text{def}}{=} \bigcup_{\delta\in\Delta} Pre[\delta](A)$ contains all *one-step predecessors*, and $Pre^*(A) \stackrel{\text{def}}{=} A \cup Pre(A) \cup Pre(Pre(A)) \cup \cdots$ all *iterated predecessors*. The successor sets $Post[\delta](A)$, $Post(A)$, and $Post^*(A)$ are defined analogously. Recall that reachability between configurations of LCS's is decidable [6, 30], which is also implied by Theorem 3.2 below.

Constrained reachability. We sometimes need to reach a set A using only rules that cannot get us out of some set $T \subseteq \mathsf{Conf}$. Formally, for $T, A \subseteq \mathsf{Conf}$, we define

$$\widehat{Pre}_T(A) \stackrel{\text{def}}{=} \{s \in \mathsf{Conf} \mid \exists\delta \in \Delta(s) \text{ s.t. } Post[\delta](s) \cap A \neq \emptyset \text{ and } Post[\delta](s) \subseteq T\}.$$

In other words, s is in $\widehat{Pre}_T(A)$ if there is a rule δ that may take s to some state in A but that cannot take it outside T. The set of iterated T-*constrained* predecessors is $\widehat{Pre}^*_T(A) \stackrel{\text{def}}{=} A \cup \widehat{Pre}_T(A) \cup \widehat{Pre}_T(\widehat{Pre}_T(A)) \cup \cdots$

3 Symbolic representations for sets of configurations

Symbolic model-checking relies on symbolic objects representing sets of configurations, and algorithmic methods for handling these objects meaningfully.

In this section, we present a symbolic framework for NPLCS's based on *differences of prefixed upward-closures*. This extends previous techniques from [4, 3, 20] in that it permits dealing with set differences and checking which is the first message in a channel. For simplicity in the presentation, *we assume that the NPLCS under consideration only has a single channel*. We also omit most of the algorithmic details pertaining to data structures, normal forms, canonization, ..., that are present in our prototype implementation (see section 6).

Recall that a set $T \subseteq \mathsf{Conf}$ is *upward-closed* (resp., *downward-closed*) if for all $s \in T$, and for all $s' \sqsupseteq s$ (resp., $s' \sqsubseteq s$), $s' \in T$. For $T \subseteq \mathsf{Conf}$, we let $\uparrow T \stackrel{\text{def}}{=} \{s \in \mathsf{Conf} | \exists s' \in T \wedge s' \sqsubseteq s\}$ denote the *upward-closure* of T, and $\downarrow T \stackrel{\text{def}}{=} \{s \in \mathsf{Conf} | \exists s' \in T \wedge s \sqsubseteq s'\}$ denote the *downward-closure* of T. For singleton sets we write shortly $\uparrow t$ and $\downarrow t$ rather than $\uparrow \{t\}$ and $\downarrow \{t\}$.

Our symbolic sets are defined with the following abstract grammar:

$$\begin{array}{rll} \text{prefix:} & \alpha := \varepsilon \mid m & m \in \mathsf{M} \\ \text{prefixed closure:} & \theta := \alpha{\uparrow}u & u \in \mathsf{M}^* \\ \text{sum of prefixed closures:} & \sigma := \theta_1 + \cdots + \theta_n & n \geq 0 \\ \text{simple symbolic set:} & \rho := \langle q, \theta - \sigma \rangle & q \in Q \text{ is a location} \\ \text{symbolic set:} & \gamma := \rho_1 + \cdots + \rho_n & n \geq 0 \end{array}$$

Prefixed (upward-)closures and their sums denote subsets of M^* defined with $[\![\alpha{\uparrow}u]\!] \stackrel{\text{def}}{=} \{\alpha v \mid u \sqsubseteq v\}$ and $[\![\theta_1 + \cdots + \theta_n]\!] \stackrel{\text{def}}{=} [\![\theta_1]\!] \cup \cdots \cup [\![\theta_n]\!]$. Symbolic sets denote subsets of Conf defined with $[\![\langle q, \theta - (\theta_1 + \cdots + \theta_n) \rangle]\!] \stackrel{\text{def}}{=} \{\langle q, v \rangle \in \mathsf{Conf} \mid v \in [\![\theta]\!] \smallsetminus ([\![\theta_1]\!] \cup \cdots \cup [\![\theta_n]\!])\}$. A *region* is any subset of Conf that can be denoted by a symbolic set. It is a *control region* if can be written under the form $\sum_i \langle q_i, \varepsilon{\uparrow}\varepsilon \rangle$, where channel contents are unrestricted.

We abuse notation and write $\emptyset$ to denote both empty (i.e., with $n = 0$) sums of prefixed closures and empty symbolic sets. We also sometimes write $\uparrow v$ for $\varepsilon{\uparrow}v$, $\theta - \theta_1 - \cdots - \theta_n$ for $\theta - (\theta_1 + \cdots + \theta_n)$, and θ for $\theta - \emptyset$. We write $\gamma \equiv \gamma'$ when $[\![\gamma]\!] = [\![\gamma']\!]$, i.e., when γ and γ' denote the same region.

Theorem 3.1 (Effective symbolic computation: basics).

Boolean closure: *Regions are closed under union, intersection, and complementation. Moreover, there exist algorithms that given symbolic sets γ_1 and γ_2 return terms denoted $\gamma_1 \sqcup \gamma_2$, $\gamma_1 \sqcap \gamma_2$ and $\neg\gamma$ such that $[\![\gamma_1 \sqcup \gamma_2]\!] = [\![\gamma_1]\!] \cup [\![\gamma_2]\!]$, $[\![\gamma_1 \sqcap \gamma_2]\!] = [\![\gamma_1]\!] \cap [\![\gamma_2]\!]$ and $[\![\neg\gamma]\!] = \mathsf{Conf} \smallsetminus [\![\gamma]\!]$.*

Upward closure: *Regions are closed under upward closure. Moreover, there exists an algorithm that given a symbolic set γ returns a term denoted $\uparrow\gamma$ such that $[\![\uparrow\gamma]\!] = \uparrow[\![\gamma]\!]$.*

Vacuity: *It is decidable whether $[\![\gamma]\!] = \emptyset$ given a region γ.*

One-step predecessors: *Regions are closed under the $Pre(\_)$ and $\widehat{Pre}\_(\_)$ operations. Moreover, there exist algorithms that given symbolic sets γ and γ' return terms denoted $Pre(\gamma)$ and $\widehat{Pre}_{\gamma'}(\gamma)$, and such that $[\![Pre(\gamma)]\!] = Pre([\![\gamma]\!])$ and $[\![\widehat{Pre}_{\gamma'}(\gamma)]\!] = \widehat{Pre}_{[\![\gamma']\!]}([\![\gamma]\!])$.*

Theorem 3.1 provides the basic ingredients necessary for symbolic model-checking of LCS's. These ingredients can then be used for computing sets defined as fixpoints.

For example, using standard μ-calculus notation, a symbolic set denoting $Pre^*([\![\gamma]\!])$ would be defined as $\mu X.\gamma \sqcup Pre(X)$. In [8] we show how a symbolic representation for sets defined by such fixpoint expressions can be computed effectively (when some guardedness condition holds).

Theorem 3.2 (Effective symbolic computation: fixpoints).

Iterated (constrained) predecessors: *Regions are closed under the* $Pre^*(\_)$ *and the* $\widehat{Pre}^*_{\_}(\_)$ *operations. Moreover, there exist algorithms that given symbolic sets* γ *and* γ' *return terms denoted* $Pre^*(\gamma)$ *and* $\widehat{Pre}^*_{\gamma'}(\gamma)$*, and such that* $[\![Pre^*(\gamma)]\!] = Pre^*([\![\gamma]\!])$ *and* $[\![\widehat{Pre}^*_{\gamma'}(\gamma)]\!] = \widehat{Pre}^*_{[\![\gamma']\!]}([\![\gamma]\!])$.

Safe sets (see section 4): *For any region* γ*, the set* $\nu X.\big(\gamma \sqcap \widehat{Pre}_X(\mathsf{Conf})\big)$ *is a region, and a term for it can be computed effectively.*

Promising sets (see section 4): *For any region* γ*, the set* $\nu X.\widehat{Pre}^*_X(\gamma)$ *is a region, and a term for it can be computed effectively.*

∃CTL: *The set of configurations satisfying an ∃CTL formula (i.e., a CTL formula where only the modalities "*$\exists(\_\ \mathrm{Until}\ \_)$*" and "*$\exists \mathrm{Next}\_$*" are allowed) is a region when the atomic propositions are themselves regions. Moreover, a symbolic set for that region can be obtained algorithmically from the ∃CTL formula.*

4 Verifying safety and liveness properties for NPLCS's

This section considers various types of safety and liveness properties where regions serve as atoms, and presents algorithms for checking the existence of a fm-scheduler $\mathcal{U}$ such that $\Pr_{\mathcal{U}}(s \models \varphi)$ is > 0, $= 1$, < 1 or $= 0$.

We start with reachability properties $\Diamond A$ and invariants $\Box A$ for some region A.

For eventually properties with the satisfaction criteria "with positive probability", decidability relies on the computation of iterative predecessors in (non-probabilistic) lossy channel systems:

Theorem 4.1. *Let* $s \in \mathsf{Conf}$ *and* $A \subseteq \mathsf{Conf}$*. There exists a scheduler* $\mathcal{U}$ *with* $\Pr_{\mathcal{U}}(s \models \Diamond A) > 0$ *iff* $\Pr_{\mathcal{U}}(s \models \Diamond A) > 0$ *for some memoryless scheduler* $\mathcal{U}$ *iff* $s \in Pre^*(A)$.

For other satisfaction criteria, or for other properties, we have to develop more ad-hoc characterizations of the sets of configurations where the qualitative properties hold.

For invariants $\Box A$, we introduce the concept of safe sets:

Definition 4.2 (Safe sets). *Let* $A, T \subseteq \mathsf{Conf}$*.* T *is called* safe *for* A *if* $T \subseteq A$ *and for all* $s \in T$*, there exists a transition rule* δ *enabled in* s *such that* $Post[\delta](s) \subseteq T$.

Since the union of safe sets is safe, the largest safe set for A, denoted $Safe(A)$, exists.

There exists a simple fixpoint characterization for $Safe(A)$ (here and in the sequel, we use the standard μ/ν-notations for fixpoints).

Lemma 4.3. *For any* $A \subseteq \mathsf{Conf}$*,* $Safe(A) = \nu X.A \cap \widehat{Pre}_X(\mathsf{Conf})$.

Thus, if A is a region, $Safe(A)$ is a region too, and a symbolic representation can be computed effectively (Theorem 3.2). This is the key for verifying invariants:

Theorem 4.4 (Safe sets and invariants). *Let* $A \subseteq \mathsf{Conf}$ *and* $s \in \mathsf{Conf}$.

(a) $s \in \mathit{Safe}(A)$
iff *there exists a scheduler* $\mathcal{U}$ *such that* $\Pr_{\mathcal{U}}(s \models \Box A) = 1$
iff *there exists a memoryless scheduler* $\mathcal{U}$ *such that* $\Pr_{\mathcal{U}}(s \models \Box A) = 1$.
(b) $s \models \exists(A \text{ Until } \mathit{Safe}(A))$
iff *there exists a scheduler* $\mathcal{U}$ *such that* $\Pr_{\mathcal{U}}(s \models \Box A) > 0$
iff *there exists a memoryless scheduler* $\mathcal{U}$ *such that* $\Pr_{\mathcal{U}}(s \models \Box A) > 0$.

The corollary is that, for a region A, we can compute a symbolic representation for the set of all configurations where $\Pr_{\mathcal{U}}(s \models \Box A) > 0$ or $= 1$ for some scheduler $\mathcal{U}$.

Definition 4.5 (Promising sets). *Let* $A, T \subseteq \mathsf{Conf}$. T *is called* promising *for* A *if for all* $s \in T$ *there exists a path* $s = s_0 \xrightarrow{\delta_1} s_1 \xrightarrow{\delta_2} \cdots \xrightarrow{\delta_m} s_m$ *with* $m \geq 0$ *such that* $s_m \in A$ *and for all* $1 \leq i \leq m$, $\mathit{Post}[\delta_i](s_{i-1}) \subseteq T$.

As for safe sets, the largest promising set for A exists: we denote it $\mathit{Prom}(A)$.

Lemma 4.6. *For any* $A \subseteq \mathsf{Conf}$, $\mathit{Prom}(A) = \nu X.\widehat{\mathit{Pre}}^{*}_{X}(A)$.

Thus, if A is a region, $\mathit{Prom}(A)$ is a region too, and a symbolic representation can be computed effectively (Theorem 3.2).

Theorem 4.7 (Promising sets and almost sure reachability). *Let* $s \in \mathsf{Conf}$ *and* $A \subseteq \mathsf{Conf}$. $s \in \mathit{Prom}(A)$ *iff* $\Pr_{\mathcal{U}}(s \models \Diamond A) = 1$ *for some scheduler* $\mathcal{U}$ *iff* $\Pr_{\mathcal{U}}(s \models \Diamond A) = 1$ *for some memoryless scheduler* $\mathcal{U}$.

The corollary is that, for a region A, we can compute the set of all configurations s such that $\Pr_{\mathcal{U}}(s \models \Diamond A) > 0$ or $= 1$ for some $\mathcal{U}$.

We now consider repeated reachability and persistence properties. The question whether a repeated reachability property $\Box\Diamond A$ holds under some scheduler with positive probability is undecidable when ranging over the full class of schedulers, but is decidable for the class of fm-schedulers. This was shown in [14, 9] for the case where A is a set of locations (*i.e.* a control region). We now show that the decidability even holds if A is a region. More precisely, we show that if A is a region and $\varphi \in \{\Box\Diamond A, \Diamond\Box A\}$, then the set of configurations s where $\Pr_{\mathcal{U}}(s \models \varphi) > 0$ or $= 1$ for some fm-scheduler is a region.

For $A \subseteq \mathsf{Conf}$ let $\mathit{Prom}^{\geq 1}(A)$ denote the largest set T of configurations such that for all $t \in T$ there exists a finite path $s = s_0 \xrightarrow{\delta_1} s_1 \xrightarrow{\delta_2} \cdots \xrightarrow{\delta_m} s_m$ with $m \geq 1$, $s_m \in A$ and $\mathit{Post}[\delta_i](s_{i-1}) \subseteq T$ for all $1 \leq i \leq m$. Note that the definition of $\mathit{Prom}^{\geq 1}(A)$ is different from $\mathit{Prom}(A)$ since the paths must have length at least 1. We then have $\mathit{Prom}^{\geq 1}(A) = \nu X.\widehat{\mathit{Pre}}^{+}_{X}(A)$, and, if A is a region then so is $\mathit{Prom}^{\geq 1}(A)$. Thus, the following theorem provides the decidability of repeated reachability and persistence properties:

Theorem 4.8 (Repeated reachability and persistence). *Let* $s \in \mathsf{Conf}$ *and* $A \subseteq \mathsf{Conf}$.

(a) $s \in Prom^{\geq 1}(A)$ *iff* $\Pr_{\mathcal{U}}(s \models \Box\Diamond A) = 1$ *for some scheduler* $\mathcal{U}$
iff $\Pr_{\mathcal{U}}(s \models \Box\Diamond A) = 1$ *for some memoryless scheduler* $\mathcal{U}$.
(b) $s \in Pre^*(Prom^{\geq 1}(A))$ *iff* $\Pr_{\mathcal{U}}(s \models \Box\Diamond A) > 0$ *for some fm-scheduler* $\mathcal{U}$
iff $\Pr_{\mathcal{U}}(s \models \Box\Diamond A) > 0$ *for some memoryless scheduler* $\mathcal{U}$.
(c) $s \in Prom(Safe(A))$ *iff* $\Pr_{\mathcal{U}}(s \models \Diamond\Box A) = 1$ *for some scheduler* $\mathcal{U}$
iff $\Pr_{\mathcal{U}}(s \models \Diamond\Box A) = 1$ *for some memoryless scheduler* $\mathcal{U}$.
(d) $s \in Pre^*(Safe(A))$ *iff* $\Pr_{\mathcal{U}}(s \models \Diamond\Box A) > 0$ *for some scheduler* $\mathcal{U}$
iff $\Pr_{\mathcal{U}}(s \models \Diamond\Box A) > 0$ *for some memoryless scheduler* $\mathcal{U}$.

We now consider the Streett formula $\varphi_S = \bigwedge_{1 \leq i \leq n} \Box\Diamond A_i \to \Box\Diamond B_i$ where $A_1, \dots, A_n$ and $B_1, \dots, B_n$ are regions. Here again we only consider fm-schedulers since the problem is undecidable for the full class of schedulers [9].

For $A, B \subseteq \mathsf{Conf}$, let $Prom_A^{\geq 1}(B)$ be the largest subset T of A such that for all $t \in T$ there exists a path $t = s_0 \xrightarrow{\delta_1} \cdots \xrightarrow{\delta_m} s_m$ with $m > 0$, $s_m \in B$ and $Post[\delta_i](s_{i-1}) \subseteq T$ for all $1 \leq i \leq m$. We have $Prom_A^{\geq 1}(B) = \nu X.\widehat{Pre}_X^{+}(B) \cap A$ and if A, B are regions then so is $Prom_A^{\geq 1}(B)$. In addition, $s \in Prom_A^{\geq 1}(B)$ iff $\Pr_{\mathcal{U}}(s \models \Box\Diamond B \wedge \Box A) = 1$ for some fm-scheduler $\mathcal{U}$.

The above is useful to show decidability of the questions whether $\Pr_{\mathcal{U}}(s \models \varphi_S) < 1$ or $= 0$ for some fm-scheduler $\mathcal{U}$. For this, we use the fact that $\Pr_{\mathcal{U}}(s \models \varphi_S) < 1$ iff $\Pr_{\mathcal{U}}(s \models \Box\Diamond A_i \to \Box\Diamond B_i) < 1$ for some i iff $\Pr_{\mathcal{U}}(s \models \Box\Diamond A_i \wedge \Diamond\Box\neg B_i) > 0$ for some i.

Theorem 4.9 (Streett property, probability less than 1). *There exists a fm-scheduler* $\mathcal{U}$ *with* $\Pr_{\mathcal{U}}(s \models \Box\Diamond A \wedge \Diamond\Box\neg B) > 0$ *iff there exists a memoryless scheduler* $\mathcal{U}$ *with* $\Pr_{\mathcal{U}}(s \models \Box\Diamond A \wedge \Diamond\Box\neg B) > 0$ *iff* $s \in Pre^*(Prom_{\neg B}^{\geq 1}(A))$. *In particular,* $\Pr_{\mathcal{U}}(s \models \varphi_S) < 1$ *for some fm-scheduler* $\mathcal{U}$ *iff* $s \in \bigcup_{1 \leq i \leq n} Pre^*(Prom_{\neg B_i}^{\geq 1}(A_i))$.

Let T_i be the set of all configurations $t \in \mathsf{Conf}$ such that $\Pr_{\mathcal{W}}(s \models \Box\Diamond A_i \wedge \Diamond\Box\neg B_i) = 1$ for some fm-scheduler $\mathcal{W}$. Note that $T_i = Pre^*(Prom_{\neg B_i}^{\geq 1}(A_i))$ is a region. Thus, $T_S = T_1 \cup T_2 \cup \cdots \cup T_n$ is a region too. This and the following theorem yields the decidability of the question whether $\Pr_{\mathcal{U}}(s \models \varphi_S) = 0$ for some scheduler $\mathcal{U}$.

Theorem 4.10 (Streett property, zero probability). *There exists a fm-scheduler* $\mathcal{U}$ *such that* $\Pr_{\mathcal{U}}\big(s \models \varphi_S\big) = 0$ *if and only if* $s \in Prom(T_S)$.

We next consider the satisfaction criterion "with positive probability". The treatment of the special case of a single strong fairness formula $\Box\Diamond A \to \Box\Diamond B \equiv \Diamond\Box\neg A \vee \Box\Diamond B$ is obvious as we have: There exists a finite-memory (resp. memoryless) scheduler $\mathcal{U}$ such that $\Pr_{\mathcal{U}}(s \models \Box\Diamond A \to \Box\Diamond B) > 0$ iff at least one of the following conditions holds: (i) there exists a fm-scheduler V such that $\Pr_{\mathcal{V}}(s \models \Diamond\Box\neg A) > 0$ or (ii) there exists a fm-scheduler $\mathcal{W}$ such that $\Pr_{\mathcal{W}}(s \models \Box\Diamond B) > 0$. We now extend this observation to the general case (several Streett properties). For $I \subseteq \{1, \dots, n\}$, let A_I denote the set of configurations s such that there exists a finite-memory scheduler satisfying $\Pr_{\mathcal{U}}(s \models \bigwedge_{i \in I} \Box\Diamond B_i \wedge \bigwedge_{i \notin I} \Box\neg A_i) = 1$ and let A be the union of all A_I's, i.e., $A = \bigcup_{I \subseteq \{1,\dots,n\}} A_I$. Then, the sets A_I and A are regions. Thus, the algorithmic treatment of Streett properties the satisfaction criteria "positive probability" and "almost surely" relies on the following theorem:

Theorem 4.11 (Streett properties, positive probability and almost surely).

(a) *There exists a fm-scheduler* $\mathcal{U}$ *such that* $\Pr_{\mathcal{U}}(s \models \varphi_S) > 0$ *iff* $s \in Pre^*(A)$.
(b) *There exists a fm-scheduler* $\mathcal{U}$ *such that* $\Pr_{\mathcal{U}}(s \models \varphi_S) = 1$ *iff* $s \in Prom(A)$.

We conclude with the following main theorem gathering all previous results:

Theorem 4.12 (Qualitative model-checking). *For any NPLCS* $\mathcal{N}$ *and Streett property* $\varphi = \bigwedge_i \Box\Diamond A_i \to \Box\Diamond B_i$ *where the* A_i*'s and* B_i*'s are regions, the set of all configurations* s *s.t. for all fm-schedulers* $\mathcal{U}$ $\Pr_{\mathcal{U}}(s \models \varphi)$ *satisfies a qualitative constraint "*$= 1$*", or "*< 1*", or "*$= 0$*", or "*> 0*", is a region that can be computed effectively.*

With the techniques of [9, § 7], Theorem 4.12 extends to all ω-regulars properties

5 Verification under fair finite-memory schedulers

We now address the problem of verifying qualitative linear time properties under fairness assumptions. Following the approaches of [19, 32, 12], we consider here a notion of *scheduler-fairness* which rules out some schedulers that generate unfair paths with positive probability. This notion of scheduler-fairness has to be contrasted with extreme- and alpha-fairness introduced in [24–26] which require a "fair" resolution of probabilistic choices and serve as verification techniques rather than fairness assumptions about the nondeterministic choices.

A scheduler $\mathcal{U}$ is called *fair* if it generates almost surely fair paths, according to some appropriate fairness constraints for paths. We deal here with *strong fairness* for selected sets of transition rules. I.e., we assume a set $\mathcal{F} = \{f_0, \dots, f_{k-1}\}$ where $f_i \subseteq \Delta$ and require strong fairness for all f_i's. (The latter means whenever some transition rule in f_i is enabled infinitely often then some transition rule in f_i will fire infinitely often.) For instance, process fairness for k processes $P_0, \dots, P_{k-1}$ can be modelled by $\mathcal{F} = \{f_0, \dots, f_{k-1}\}$ where f_i is the set of transition rules describing P_i's actions.

A set $f \subseteq \Delta$ is called enabled in configuration s if there is a transition rule $\delta \in f$ that is enabled in s, i.e., if $\Delta(s) \cap f \neq \emptyset$. If F is a subset of $\mathcal{F}$ and $s \in \mathsf{Conf}$ then F is called enabled in s if some $f \in F$ is enabled in s, i.e., if $\exists f \in F. f \cap \Delta(s) \neq \emptyset$. We write $\mathsf{Enabl}(F)$ to denote the set of configurations $s \in \mathsf{Conf}$ where F is enabled.

Definition 5.1 (Fair paths, fair schedulers). *Let* $\mathcal{F} \in 2^{2^\Delta}$ *be a (finite) set consisting of subsets of* Δ*. An infinite path* $s_0 \xrightarrow{\delta_1} s_1 \xrightarrow{\delta_2} \cdots$ *is called* $\mathcal{F}$-fair *iff for all* $f \in \mathcal{F}$ *either* $\delta_j \in f$ *for infinitely many* j *or there is some* $i \geq 0$ *such that* f *is not enabled in the configurations* s_j *for all* $j \geq i$*. Scheduler* $\mathcal{U}$ *is called* $\mathcal{F}$-fair *(or briefly* fair*) if for each starting state* s*, almost all* $\mathcal{U}$*-paths are* $\mathcal{F}$*-fair.*

We first consider *reachability* properties $\Diamond A$ and show that fairness assumptions are irrelevant for the satisfaction criteria "with positive probability"and "almost surely". This follows from the fact that from the moment on where a configuration in A has been entered one can continue in an arbitrary, but $\mathcal{F}$-fair way. Thus:

$$\begin{array}{lcl} \exists\, \mathcal{V}\ \mathcal{F}\text{-fair s.t. } \Pr_{\mathcal{V}}(s \models \Diamond A) > 0 & \text{iff} & \exists\, \mathcal{U} \text{ s.t. } \Pr_{\mathcal{U}}(s \models \Diamond A) > 0 \\ \exists\, \mathcal{V}\ \mathcal{F}\text{-fair s.t. } \Pr_{\mathcal{V}}(s \models \Diamond A) = 1 & \text{iff} & \exists\, \mathcal{U} \text{ s.t. } \Pr_{\mathcal{U}}(s \models \Diamond A) = 1 \end{array}$$

By the results of section 4, given an NPLCS $\mathcal{N}$, starting configuration s and region A, the questions whether there exists a $\mathcal{F}$-fair scheduler $\mathcal{U}$ such that $\Pr_{\mathcal{U}}(s \models \Diamond A) > 0$ or $= 1$ are decidable.

The treatment of *invariant* properties $\Box A$ under fairness constraints relies on generalizations of the concept of safe and promising sets. For $A, B \subseteq \mathsf{Conf}$, $Prom_A(B)$ denotes the largest set $T \subseteq A \cup B$ such that for all $t \in T$ there exists a path $t = s_0 \xrightarrow{\delta_1} \cdots \xrightarrow{\delta_m} s_m$ with $m \geq 0$, $s_m \in B$ and $Post[\delta_i](s_{i-1}) \subseteq T$ for all $1 \leq i \leq m$. The fixed-point definition of $Prom_A(B)$ would be $\nu X.\widehat{Pre}_X^*(B) \cap (A \cup B)$.

For $\mathcal{F} \subseteq 2^\Delta$ and $A \subseteq \mathsf{Conf}$, let $Safe_{\mathcal{F}}(A) = \bigcup_{F \subseteq \mathcal{F}} Safe[F](A)$ where $Safe[F](A)$ is defined as follows. If F is a nonempty subset of $\mathcal{F}$ then $Safe[F](A)$ denotes the largest set $T \subseteq A \setminus \mathsf{Enabl}(\mathcal{F} \setminus F)$ such that for all $t \in T$ and $f \in F$ there is a path $s_0 \xrightarrow{\delta_1} \cdots \xrightarrow{\delta_m} s_m$ with $t = s_0$, $m \geq 1$, $\delta_m \in f$ and $Post[\delta_i](s_{i-1}) \subseteq T$ for all $1 \leq i \leq m$. Moreover, $Safe_{\mathcal{F}}[\emptyset](A) = Safe(A \setminus \mathsf{Enabl}(\mathcal{F}))$.

Since $\mathsf{Enabl}(\mathcal{F} \setminus F)$ can be expressed by $Pre[\mathcal{F} \setminus F](\mathsf{Conf})$, we get the following mu-calculus terms for $Safe[\emptyset](A)$ and $Safe[F](A)$:

- $Safe[\emptyset](A) = \nu X.\big(A \setminus Pre[\mathcal{F}](\mathsf{Conf})\big) \cap \widehat{Pre}_X(\mathsf{Conf})$, and
- $Safe[F](A) = \nu X.\big(A \setminus Pre[\mathcal{F} \setminus F](\mathsf{Conf})\big) \cap \bigcap_{f \in F} \widehat{Pre}_X^*(\widehat{Pre}_X[f](\mathsf{Conf}))$.

Theorem 5.2 (Fair invariants). *Let $A \subseteq \mathsf{Conf}$ and $s \in \mathsf{Conf}$.*

(a) There is a $\mathcal{F}$-fair fm-scheduler $\mathcal{V}$ s.t. $\Pr_{\mathcal{V}}(s \models \Box A) > 0$ *iff* $s \models \exists(A \,\mathrm{Until}\, Safe_{\mathcal{F}}(A))$.
(b) There is a $\mathcal{F}$-fair fm-scheduler $\mathcal{V}$ s.t. $\Pr_{\mathcal{V}}(s \models \Box A) = 1$ *iff* $s \in Prom_A(Safe_{\mathcal{F}}(A))$.

Observe that, for a region γ, $Safe_{\mathcal{F}}([\![\gamma]\!])$ and $Prom_A(Safe_{\mathcal{F}}([\![\gamma]\!]))$ are regions that can be built effectively (based on the same reasoning that we use for Theorem 3.2). Thus, Theorem 5.2 yields the decidability of the questions whether for a given NPLCS, region A and configuration s, there exists a $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$ such that $\Pr_{\mathcal{U}}(s \models \Box A) > 0$ or $= 1$.

In the sequel, for $A \subseteq \mathsf{Conf}$, we denote by $T^{\mathcal{F}}_{\Box A}$ the set of all configurations s such that $\Pr_{\mathcal{U}}(s \models \Box A) = 1$ for some $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$.

We now come to *repeated reachability* $\Box\Diamond A$ and *persistence* $\Diamond\Box A$ properties under fairness constraints. For $A \subseteq \mathsf{Conf}$, we define $T^{\mathcal{F}}_{\Box\Diamond A} = \bigcup_{F \subseteq \mathcal{F}} T_F$ where T_F is the largest subset of $\mathsf{Conf} \setminus \mathsf{Enabl}(\mathcal{F} \setminus F)$ such that for all $t \in T_F$:

- there is a finite path $s_0 \xrightarrow{\delta_1} \cdots \xrightarrow{\delta_m} s_m$ with $m \geq 1$, $t = s_0$, $s_m \in A$ and $Post[\delta_i](s_{i-1}) \subseteq T_F$ for all $1 \leq i \leq m$,
- for each $f \in F$ there is a finite path $s_0 \xrightarrow{\delta_1} \cdots \xrightarrow{\delta_m} s_m$ with $t = s_0$, $m \geq 1$, $\delta_m \in f$ and $Post[\delta_i](s_{i-1}) \subseteq T_F$ for all $1 \leq i \leq m$.

Theorem 5.3 (Fair repeated reachability and persistence). *Let $A \subseteq \mathsf{Conf}$ and $s \in \mathsf{Conf}$.*

(a) There exists a $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$ with $\Pr_{\mathcal{U}}(s \models \Box\Diamond A) = 1$ *iff* $s \in Prom(T^{\mathcal{F}}_{\Box\Diamond A})$.
(b) There exists a $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$ with $\Pr_{\mathcal{U}}(s \models \Box\Diamond A) > 0$ *iff* $s \in Pre^*(T^{\mathcal{F}}_{\Box\Diamond A})$.
(c) There exists a $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$ with $\Pr_{\mathcal{U}}(s \models \Diamond\Box A) = 1$ *iff* $s \in Prom(T^{\mathcal{F}}_{\Box A})$.

(d) *There exists a $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$ with* $\Pr_{\mathcal{U}}(s \models \Diamond\Box A) > 0$ *iff* $s \in Pre^*(T^{\mathcal{F}}_{\Box A})$.

With similar arguments as for *Prom*(A), the sets of configuration $T^{\mathcal{F}}_{\Box\Diamond A}$ and $T^{\mathcal{F}}_{\Box A} = Prom_A(Safe_{\mathcal{F}}(A))$ are regions whenever A is a region. This entails the decidability of the questions whether given region A, there exists a $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$ such that $\Pr_{\mathcal{U}}(s \models \varphi) = 1$ or > 0 where $\varphi = \Box\Diamond A$ or $\Diamond\Box A$.

We next consider *linear time* properties, formalized by LTL formulas φ where regions serve as atomic propositions. The idea is to encode the fairness constraints in the model (the NPLCS) by a Streett property

$$fair = \bigwedge_{f \in \mathcal{F}} (\Box\Diamond A_f \to \Box\Diamond B_f)$$

(with regions $A_f, B_f \subseteq \mathsf{Conf}$) that will be considered in conjunction with φ. We modify the given LCS $\mathcal{L} = (Q, \mathsf{C}, \mathsf{M}, \Delta)$ and construct a new LCS $\mathcal{L}' = (Q', \mathsf{C}, \mathsf{M}, \Delta')$ as follows. We introduce new locations q_F for all subsets F of $\mathcal{F}$ and $q \in Q$, i.e., we deal with $Q' = \{q_F : q \in Q, F \subseteq \mathcal{F}\}$. Δ' is the smallest set of transition rules such that $p_G \xrightarrow{op} q_F \in \Delta'$ if $p \xrightarrow{op} q \in \Delta$, $G \subseteq \mathcal{F}$ and $F = \{f \in \mathcal{F} : p \xrightarrow{op} q \in f\}$. For $f \in \mathcal{F}$, B_f is the set of configurations $\langle q_F, w\rangle$ in $\mathcal{L}'$ such that $f \in F$, while A_f denotes the set of all configurations $\langle q_F, w\rangle$ of $\mathcal{L}'$ where f is enabled in the configuration $\langle q, w\rangle$ of $\mathcal{L}$. We finally transform the given formula φ into φ' by replacing any region C of $\mathcal{L}$ that appears as an atom in φ with the region $C' = \{\langle q_F, w\rangle : \langle q, w\rangle \in C, F \subseteq \mathcal{F}\}$. For instance, if $\varphi = \Box\Diamond(q \wedge (c \neq \varepsilon))$ then $\varphi' = \Box\Diamond\big((q \vee \bigvee_{F \subseteq \mathcal{F}} q_F) \wedge (c \neq \varepsilon)\big)$.

In the sequel, let $\mathcal{N} = (\mathcal{L}, \tau)$ be the NPLCS that we want to verify against φ and let $\mathcal{N}' = (\mathcal{L}', \tau)$ the associated modified NPLCS. Obviously, for each fm-scheduler $\mathcal{U}$ for $\mathcal{N}$ there is a "corresponding" fm-scheduler $\mathcal{U}'$ for $\mathcal{N}'$, and vice versa. Corresponding means that $\mathcal{U}'$ behaves as $\mathcal{U}$ for the current configuration $\langle q, w\rangle$ with $q \in Q$. If the current configuration of $\mathcal{U}'$ is $\langle q_F, w\rangle$ then $\mathcal{U}'$ behaves as $\mathcal{U}$ for $\langle q, w\rangle$. Then, $\Pr_{\mathcal{U}}(s \models \varphi) = \Pr_{\mathcal{U}'}(s \models \varphi')$ for all configurations s in $\mathcal{N}$. Here, each configuration $s = \langle q, w\rangle$ of $\mathcal{N}$ is identified with the configuration $\langle q_\emptyset, w\rangle$ in $\mathcal{N}'$. Moreover, $\mathcal{U}$ is $\mathcal{F}$-fair iff $\Pr_{\mathcal{U}'}(s \models fair) = 1$. This yields part (a) of the following lemma. Part (b) follows from the fact that $\Pr_{\mathcal{U}}(s \models \varphi) = 1 - \Pr_{\mathcal{U}}(s \models \neg\varphi)$ for each scheduler $\mathcal{U}$.

Lemma 5.4. *Let s be a configuration in $\mathcal{N}$ (and $\mathcal{N}'$) and φ an LTL formula. Then:*

(a) *There exists a $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$ for $\mathcal{N}$ such that* $\Pr_{\mathcal{U}}(s \models \varphi) = 1$ *if and only if there exists a fm-scheduler $\mathcal{U}'$ for $\mathcal{N}'$ such that* $\Pr_{\mathcal{U}'}(s \models fair \wedge \varphi') = 1$.
(b) *There exists a $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$ for $\mathcal{N}$ such that* $\Pr_{\mathcal{U}}(s \models \varphi) = 0$ *if and only if there exists a fm-scheduler $\mathcal{V}$ for $\mathcal{N}'$ such that* $\Pr_{\mathcal{V}}(s \models fair \wedge \neg\varphi') = 1$.
(c) *There exists a $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$ for $\mathcal{N}$ such that* $\Pr_{\mathcal{U}}(s \models \varphi) > 0$ *if and only if there exists a fm-scheduler $\mathcal{V}$ for $\mathcal{N}'$ such that* $\Pr_{\mathcal{V}}(s \models fair \wedge \varphi') > 0$.
(d) *There exists a $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$ for $\mathcal{N}$ such that* $\Pr_{\mathcal{U}}(s \models \varphi) < 1$ *if and only if there exists a fm-scheduler $\mathcal{V}$ for $\mathcal{N}'$ such that* $\Pr_{\mathcal{V}}(s \models fair \wedge \neg\varphi') > 0$.

Lemma 5.4 even holds for arbitrary ω-regular properties. It provides a reduction from the verification problem for qualitative LTL formulas in NPLCS's and fair fm-schedulers to the same problem for the full class of fm-schedulers. Thus, all decidability results that have been established for NPLCS's and qualitative verification problems for the class of fm-schedulers (see 4) also hold when fairness assumptions are made.

6 Automatic verification of Pachl's protocol

Fig. 1 directly translates into a LCS $\mathcal{L}_{\text{Pachl}}$ when the asynchronous product of P_{L} and P_{R} is considered. $\mathcal{L}_{\text{Pachl}}$ has $6 \times 6 = 36$ control locations and $(18+18) \times 6 = 216$ transition rules. In order to reason about notions like "a rule δ has been fired", that are ubiquitous in fairness hypothesis, our tool adds an history variable recording the last fired rule (actually, only its action label). This would further multiply the number of states and of transitions by 20, but not all pairs (location,last action) are meaningful so that the final model can be stripped down to 144 locations and 948 rules. In all our results below we do not use the names of these 144 locations, but rather project them to the more readable underlying 36 locations.

6.1 Safety analysis

Pachl [22] computed manually the set $Post^*(\texttt{Init})$ of all configurations reachable in $\mathcal{L}_{\text{Pachl}}$ from the initial empty configuration $\texttt{Init} = (\texttt{L0}, \texttt{R4}, \varepsilon, \varepsilon)$, and such forward computations can sometimes be done automatically with the techniques described in [4] (although termination of the forward-reachability computations cannot be guaranteed in general). These computations show that the protocol does indeed preserve the integrity of communication in the sense that no confusion between data messages is introduced by losses.

Our calculus for regions is geared towards backward computation, where termination is guaranteed. Our implementation can compute automatically the set of deadlock configurations:

$$\texttt{Dead} \stackrel{\text{def}}{=} \mathsf{Conf} \smallsetminus Pre(\mathsf{Conf}) \;=\; \langle \texttt{L4}, \texttt{R4}, \varepsilon, \varepsilon \rangle.$$

Hopefully, $\texttt{Dead}$ is not reachable from $\texttt{Init}$. We can compute the set $Pre^*(\texttt{Dead})$ of all unsafe configurations, that can end up in a deadlock. Intersecting with $\uparrow\texttt{Init}$, we obtain the set of unsafe starting channel contents:

$$\begin{aligned} &Pre^*(\texttt{Dead}) \sqcap \uparrow\texttt{Init} \;= \\ &\qquad\qquad \langle \texttt{L0}, \texttt{R4}, \uparrow\varepsilon, \uparrow \mathrm{a}_0\mathrm{d}_0 \rangle \;+\; \langle \texttt{L0}, \texttt{R4}, \uparrow \mathrm{eod}\,\mathrm{a}_0, \uparrow \mathrm{a}_0 \rangle \;+\; \langle \texttt{L0}, \texttt{R4}, \uparrow \mathrm{d}_0\mathrm{eod}\,\mathrm{a}_0, \uparrow\varepsilon \rangle. \end{aligned}$$

Thus eventual deadlock is possible from location $(\texttt{L0}, \texttt{R4})$ if the channels initially contain the appropriately unsafe contents.

6.2 Liveness analysis

We now come to what is the main motivation of our work: proving progress under fairness hypothesis. In this case study, the problem we address is in general to compute the set of all configurations satisfying some $\Pr_{\mathcal{U}}(s \models \Box\Diamond A) = 1$ for all schedulers $\mathcal{U}$ satisfying some fairness conditions $\mathcal{F}$. Following equivalences of section 5, this is related to the computation of $T^{\mathcal{F}}_{\Box\Diamond A}$. More precisely: $\{s | \forall \mathcal{U}\, \mathcal{F}\text{-fair } \Pr_{\mathcal{U}}(s \models \Box\Diamond A) = 1\} = \mathsf{Conf} \smallsetminus Pre^*(T^{\mathcal{F}}_{\Box\Diamond A})$.

When computing $T^{\mathcal{F}}_{\Box\Diamond A}$, all subsets of $\mathcal{F}$ have to be considered and this induces a combinatorial explosion for large $\mathcal{F}$. Since we did not yet develop and implement heuristics to overcome this difficulty, we only checked examples considering "small" $\mathcal{F}$ sets (meaning a number of fairness sets, each of which can be a large set of rules) in this preliminary study. For example, we considered "strong process fairness" $\mathcal{F}_{\texttt{process}} = \{F_{\texttt{left}}, F_{\texttt{right}}\}$ (with obvious meaning for the sets of transitions $F_{\texttt{left}}, F_{\texttt{right}}$), or "strong fairness for reading" $\mathcal{F}_{\texttt{read}} = \{F_{\texttt{read}}\}$.

Regarding the target set A, we consider questions whether a given transition (in P_{L} or P_{R}) is fired infinitely often (using the history variable), or whether a process changes control states infinitely often, etc. Observe that a conjunction of "$\Pr_{\mathcal{U}}(s \models \Box\Diamond A_i) = 1$" gives $\Pr_{\mathcal{U}}(s \models \bigwedge_i \Box\Diamond A_i) = 1$, so that we can check formulas like $\bigwedge_i \Box\Diamond\texttt{Li} \wedge \bigwedge_i \Box\Diamond\texttt{Ri}$, expressing progress in communication between the two processes.

In the three following cases :

- $\mathcal{F} = \mathcal{F}_{\texttt{read}}$ and $A = \mathit{After}_{\texttt{left}}$
- $\mathcal{F} = \mathcal{F}_{\texttt{read}}$ and $A = \mathit{After}_{\texttt{left-move}}$
- $\mathcal{F} = \{F_{\texttt{read}}, F_{\texttt{right-read}}\}$ and $A = \mathit{After}_{\texttt{left}}$

our prototype model checker yields that $\texttt{Init} \in \mathsf{Conf} \setminus \mathit{Pre}^*(T^{\mathcal{F}}_{\Box\Diamond A})$. This means that, in all three cases, starting from $\texttt{Init}$, the set of configurations A will be visited infinitely often almost surely, under all $\mathcal{F}$-fair schedulers.

7 Conclusion

We introduced NPLCS's, a model for nondeterministic channel systems where messages are lost probabilistically, and showed the decidability of qualitative verification question of the form "does φ holds with probability 1 for all $\mathcal{F}$-fair finite-memory schedulers?" where φ is an ω-regular linear-time property and $\mathcal{F}$ a strong fairness condition.

When atomic propositions can refer to the contents of channels, which is required when one wants to express fairness and firability of rules, our decidability results rest upon a new notion of symbolic regions based on "prefixed upward-closures". These symbolic methods can be implemented rather directly and we used them to analyze simple systems.

These results are the outcome of a research project that started in [13, 14] with the first early definition of NPLCS's and was continued in [9] where the key notions for reducing to constrained reachability questions have been first identified in a simplified framework. Further developments will focus on incorporating algorithmic ideas from symbolic verification (normal forms, caches, sharing, ...) in our naive prototype verifier, turning it into a more solid analysis tool.

References

1. P. A. Abdulla, C. Baier, S. Purushothaman Iyer, and B. Jonsson. Simulating perfect channels with probabilistic lossy channels. *Information and Computation*, 197(1–2):22–40, 2005.

2. P. A. Abdulla, N. Bertrand, A. Rabinovich, and Ph Schnoebelen. Verification of probabilistic systems with faulty communication. *Information and Computation*, 202(2):141–165, 2005.
3. P. A. Abdulla, A. Bouajjani, and J. d'Orso. Deciding monotonic games. In *Proc. 17th Int. Workshop Computer Science Logic (CSL 2003) and 8th Kurt Gödel Coll. (KGL 2003), Vienna, Austria, Aug. 2003*, volume 2803 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2003.
4. P. A. Abdulla, A. Collomb-Annichini, A. Bouajjani, and B. Jonsson. Using forward reachability analysis for verification of lossy channel systems. *Formal Methods in System Design*, 25(1):39–65, 2004.
5. P. A. Abdulla and B. Jonsson. Undecidable verification problems for programs with unreliable channels. *Information and Computation*, 130(1):71–90, 1996.
6. P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996.
7. C. Baier, N. Bertrand, and Ph. Schnoebelen. A note on the attractor-property of infinite-state Markov chains. *Information Processing Letters*, 97(2):58–63, 2006.
8. C. Baier, N. Bertrand, and Ph. Schnoebelen. On computing fixpoints in well-structured regular model checking, with applications to lossy channel systems. RR cs.CS/0606091, Computing Research Repository, June 2006. Visible at http://arxiv.org/abs/cs.CS/0606091.
9. C. Baier, N. Bertrand, and Ph. Schnoebelen. Verifying nondeterministic probabilistic channel systems against ω-regular linear-time properties. RR cs.LO/0511023, Computing Research Repository, April 2006. To be published in *ACM Trans. Computational Logic*, visible at http://arxiv.org/abs/cs.LO/0511023.
10. 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.
11. C. Baier, B. R. Haverkort, H. Hermanns, J.-P. Katoen, and M. Siegle, editors. *Validation of Stochastic Systems – A Guide to Current Research*, volume 2925 of *Lecture Notes in Computer Science*. Springer, 2004.
12. C. Baier and M. Kwiatkowska. Model checking for a probabilistic branching time logic with fairness. *Distributed Computing*, 11(3):125–155, 1998.
13. 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.
14. N. Bertrand and Ph. Schnoebelen. Verifying nondeterministic channel systems with probabilistic message losses. In Ramesh Bharadwaj, editor, *Proc. 3rd Int. Workshop on Automated Verification of Infinite-State Systems (AVIS 2004), Barcelona, Spain, Apr. 2004*, 2004.
15. D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2):323–342, 1983.
16. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.
17. A. Finkel. Decidability of the termination problem for completely specificied protocols. *Distributed Computing*, 7(3):129–135, 1994.
18. B. Hailpern and S. Owicki. Verifying network protocols using temporal logic. In *Proc. NBS/IEEE Symposium on Trends and Applications 1980: Computer Network Protocols, Gaithersburg, MD, May 1980*, pages 18–28. IEEE Comp. Soc. Press, 1980.
19. S. Hart, M. Sharir, and A. Pnueli. Termination of probabilistic concurrent programs. *ACM Transactions on Programming Languages and Systems*, 5(3):356–380, 1983.
20. A. Kučera and Ph. Schnoebelen. A general approach to comparing infinite-state systems with their finite-state specifications. *Theoretical Computer Science*, 2006. To appear.

21. 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.
22. J. K. Pachl. Protocol description and analysis based on a state transition model with channel expressions. In *Proc. 7th IFIP WG6.1 Int. Workshop on Protocol Specification, Testing, and Verification (PSTV '87), Zurich, Switzerland, May 1987*, pages 207–219. North-Holland, 1987.
23. P. Panangaden. Measure and probability for concurrency theorists. *Theoretical Computer Science*, 253(2):287–309, 2001.
24. A. Pnueli. On the extremely fair treatment of probabilistic algorithms. In *Proc. 15th ACM Symp. Theory of Computing (STOC '83), Boston, MA, Apr. 1983*, pages 278–290. ACM Press, 1983.
25. A. Pnueli and L. D. Zuck. Verification of multiprocess probabilistic protocols. *Distributed Computing*, 1(1):53–72, 1986.
26. A. Pnueli and L. D. Zuck. Probabilistic verification. *Information and Computation*, 103(1):1–29, 1993.
27. 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.
28. M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 1994.
29. 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.
30. Ph. Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letters*, 83(5):251–261, 2002.
31. Ph. Schnoebelen. The verification of probabilistic lossy channel systems. In Baier et al. [11], pages 445–465.
32. M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proc. 26th IEEE Symp. Foundations of Computer Science (FOCS '85), Portland, OR, USA, Oct. 1985*, pages 327–338. IEEE Comp. Soc. Press, 1985.

A Proofs for section 3

A.1 Proof of Theorem 3.1

Union: Clearly $\gamma_1 + \gamma_2$ can be taken for $\gamma_1 \sqcup \gamma_2$.

Intersection: We consider simpler terms first. For two upward closures $\uparrow u$ and $\uparrow v$, the intersection $\uparrow u \sqcap \uparrow v$ is the sum $\tau = \sum_{i=1}^{n} \uparrow w_i$ where the w_i's are all the minimal words in $[\![\uparrow u]\!] \cap [\![\uparrow v]\!]$. For example, $\uparrow 010 \sqcap \uparrow 210 = \uparrow 2010 + \uparrow 01201 + \uparrow 02101$. The w_i's can be computed effectively since the length of any w_i is bounded by $|u| + |v|$. Observe that $n \leq \binom{|u|+|v|}{|u|} = \binom{|u|+|v|}{|v|}$ and this bound is reached, e.g., when u and v share no messages.

For prefixed upward-closures and later constructs, we introduce two auxiliary operations: taking the left residual, $m^{-1}u$, or the right residual, $u\,m^{-1}$, of a word $u \in \mathsf{M}^*$ by a message $m \in \mathsf{M}$:

$$m^{-1}u \stackrel{\text{def}}{=} \begin{cases} v & \text{if } u = mv, \\ u & \text{otherwise,} \end{cases} \qquad\qquad u\,m^{-1} \stackrel{\text{def}}{=} \begin{cases} v & \text{if } u = vm, \\ u & \text{otherwise.} \end{cases}$$

Now, the intersection of two prefixed closures reduces to the previous case via:

$$\alpha{\uparrow}u \sqcap \beta{\uparrow}v \stackrel{\text{def}}{=} \begin{cases} \alpha(\uparrow u \sqcap \uparrow v) & \text{if } \alpha = \beta, \\ \alpha(\uparrow u \sqcap \uparrow(\alpha^{-1}v)) & \text{if } \alpha \in \mathsf{M} \text{ and } \beta = \varepsilon, \\ \beta(\uparrow(\beta^{-1}u) \sqcap \uparrow v) & \text{if } \beta \in \mathsf{M} \text{ and } \alpha = \varepsilon, \\ \emptyset & \text{if } \alpha \neq \beta, \alpha, \beta \in \mathsf{M}, \end{cases}$$

where $\alpha(\uparrow u \sqcap \uparrow v)$ is short for $\sum_i \alpha{\uparrow}w_i$ if $\uparrow u \sqcap \uparrow v = \sum_i \uparrow w_i$. We can now use:

$$\langle q, \theta - \sigma\rangle \sqcap \langle q', \rho' - \sigma'\rangle \stackrel{\text{def}}{=} \begin{cases} \sum_i \langle q, \theta_i - \sigma - \sigma'\rangle & \text{if } q = q' \text{ and } \theta \sqcap \theta' = \sum_i \theta_i, \\ \emptyset & \text{otherwise,} \end{cases}$$

$$\sum_i \rho_i \bigsqcap \sum_j \rho_j \stackrel{\text{def}}{=} \sum_{i,j} (\rho_i \sqcap \rho_j).$$

Complementation: For simple sets we use

$$\neg\langle q, \alpha{\uparrow}v - \sum_{i=1}^{n} \beta_i{\uparrow}w_i\rangle \stackrel{\text{def}}{=} \langle q, \uparrow\varepsilon - \alpha{\uparrow}v\rangle + \sum_{i=1}^{n} \langle q, \beta_i{\uparrow}w_i\rangle + \sum_{p \neq q} \langle p, \uparrow\varepsilon\rangle.$$

With intersection, this allows complementing general sets via $\neg(\sum_i \rho_i) \stackrel{\text{def}}{=} \bigsqcap_i \neg\rho_i$.

Membership: Telling whether $\langle q, u\rangle$ belongs to some $[\![\gamma]\!]$ is obviously decidable.

Upward closure: The difficult case is the upward closure of a difference $\alpha{\uparrow}u-\sigma$. Obviously, if $\alpha u \notin [\![\sigma]\!]$, then $\uparrow(\alpha{\uparrow}u-\sigma) \equiv \uparrow(\alpha u)$. However, it is possible to have simultaneously $\alpha u \in [\![\sigma]\!]$ and $\alpha{\uparrow}u-\sigma \not\equiv \emptyset$. For example $\uparrow ab - a{\uparrow}b \equiv b{\uparrow}ab$ when $\mathsf{M}=\{a,b\}$.

We use:

$$\uparrow(\alpha{\uparrow}u-\sigma) \stackrel{\text{def}}{=} \begin{cases} \uparrow(\alpha u) & \text{if } \alpha u \notin [\![\sigma]\!], \\ \emptyset & \text{if } (\alpha \in \mathsf{M} \text{ or } \alpha u = \varepsilon) \text{ and } \alpha u \in [\![\sigma]\!], \\ \sum_{m' \neq m} \uparrow(m'{\uparrow}u-\sigma) & \text{if } \alpha=\varepsilon,\ u=mv, \text{ and } u \in [\![\sigma]\!]. \end{cases}$$

Here the third case is recursively given in terms of the first two cases but the definition is obviously well-founded.

Vacuity: Vacuity is trivially decidable for sets γ where no restrictions occur (*i.e.* sums of prefixed upward-closures). For sets with restrictions, we reduce to the simpler case with $\gamma \equiv \emptyset$ iff $\uparrow\gamma \equiv \emptyset$.

One-step predecessors: We start with sets of the form $Pre[\delta]\langle q, \uparrow v\rangle$. There are three cases, depending on the form of δ:

$$Pre[r \xrightarrow{c?m} q]\langle q, \uparrow v\rangle \stackrel{\text{def}}{=} \langle r, m{\uparrow}v\rangle, \qquad \text{(Read operation)}$$

$$Pre[r \xrightarrow{c?m} q]\langle q, \uparrow v\rangle \stackrel{\text{def}}{=} \langle r, \uparrow(v\,m^{-1})\rangle, \qquad \text{(Write operation)}$$

$$Pre[r \xrightarrow{\surd} q]\langle q, \uparrow v\rangle \stackrel{\text{def}}{=} \langle r, \uparrow v\rangle, \qquad \text{(Internal step)}$$

while $Pre[r \xrightarrow{op} p]\langle q, \uparrow v\rangle = \emptyset$ if $p \neq q$.

This is enough to compute one-step predecessors of arbitrary sets since $Pre[\delta](\bigcup S_i) = \bigcup_i Pre[\delta](S_i)$ and, $Pre[\delta](S) = Pre[\delta](\uparrow S)$ because arbitrary message losses can occur after any step.

Hence, $Pre[\delta](\gamma)$ is a sum of prefixed upward-closures (in case of a read operation) or a sum of upward-closures (in case of a write or internal operation).

Constrained predecessors: We use $\widehat{Pre}_{\gamma'}(\gamma) \stackrel{\text{def}}{=} \bigsqcup_{\delta\in\Delta}\big(Pre[\delta](\gamma) \smallsetminus Pre[\delta](\neg\gamma')\big)$.

A.2 Proof of Theorem 3.2

We first show how the sets we have to compute symbolically can be defined as upward/-downward-guarded μ-formulas. Then the result from [8] applies and provides effective computability by symbolic methods.

For Theorem 3.2, we are interested in the following sets:

$$\begin{aligned} Pre^*(\gamma) &= \mu X.\big(\gamma \sqcup Pre(X)\big), & \widehat{Pre}^*_{\gamma_1}(\gamma_2) &= \mu X.\big(\gamma_2 \sqcup \widehat{Pre}_{\gamma_1}(X)\big), \\ Safe(\gamma) &= \nu X.\big(\gamma \sqcap \widehat{Pre}_X(\mathsf{Conf})\big), & Prom(\gamma) &= \nu X.\widehat{Pre}^*_X(\gamma), \\ \exists\big(\gamma \,\mathrm{Until}\, \gamma'\big) &= \mu X.\Big(\gamma' \sqcup \big(\gamma \sqcap Pre(X)\big)\Big), & \exists\big(\mathrm{Next}\,\gamma\big) &= Pre(\gamma). \end{aligned} \qquad (*)$$

Observe that the definitions as least/greatest fixpoint expressions are well-formed since $(2^{\mathsf{Conf}}, \subseteq)$ is a complete lattice, and since only monotonic operations $(\sqcup, \sqcap, \mathit{Pre}, \ldots)$ are used in our fixpoint expressions. In particular, $\widehat{\mathit{Pre}}$ and $\widehat{\mathit{Pre}}^*$ are monotonic in both their arguments:

Fact A.1. *If $A \subseteq B$ and $T \subseteq S$, then $\widehat{\mathit{Pre}}_T(A) \subseteq \widehat{\mathit{Pre}}_S(B)$ and $\widehat{\mathit{Pre}}^*_T(A) \subseteq \widehat{\mathit{Pre}}^*_S(B)$.*

We say that a μ-calculus expression e is *upward-guarded* if every subexpression of the form $\mu X.f$ is such that all free occurrences of X in f are under the scope of an upward-closure "$\uparrow \_$" operator. We say that e is *downward-guarded* if for all subexpressions of the form $\nu X.f$ all occurrences of X in f are under the scope of a downward-closure "$\downarrow \_$" operator. Finally, e is just *guarded* if it is both upward- and downward-guarded.

In a more abstract setting, the main result of [8] just states that subsets of Conf defined by guarded expressions are regions, and that they can be computed effectively. We now prove Theorem 3.2 by showing that the expressions in (*) are guarded, or can be replaced by (equivalent) guarded expressions.

Iterated predecessors: Since $\mathit{Pre}(A) = \mathit{Pre}(\uparrow A)$ for any $A \subseteq \mathsf{Conf}$, we can rewrite the definition of $\mathit{Pre}^*(\gamma)$ as

$$\mathit{Pre}^*(\gamma) \stackrel{\text{def}}{=} \mu X.\big(\gamma \sqcup \mathit{Pre}(\uparrow X)\big)$$

which is guarded.

Iterated constrained predecessors: Since $\widehat{\mathit{Pre}}(A) = \widehat{\mathit{Pre}}(\uparrow A)$ for any $A \subseteq \mathsf{Conf}$, we can rewrite the definition of $\widehat{\mathit{Pre}}^*(\gamma)$ as

$$\widehat{\mathit{Pre}}^*_{\gamma_1}(\gamma_2) \stackrel{\text{def}}{=} \mu X.\big(\gamma_2 \sqcup \widehat{\mathit{Pre}}_{\gamma_1}(\uparrow X)\big)$$

which is guarded.

Safe sets: We start with the following lemma:

Lemma A.1. *For all $A, T \subseteq \mathsf{Conf}$, $\widehat{\mathit{Pre}}_T(A) = \widehat{\mathit{Pre}}_{K_\downarrow(T)}(A)$, and $\widehat{\mathit{Pre}}^*_T(A) = \widehat{\mathit{Pre}}^*_{K_\downarrow(T)}(A)$.*

Proof. Clearly, $K_\downarrow(T) \subseteq T$ implies $\widehat{\mathit{Pre}}_{K_\downarrow(T)}(A) \subseteq \widehat{\mathit{Pre}}_T(A)$. Let $s \in \widehat{\mathit{Pre}}_T(A)$. There exists δ enabled in s such that $\mathit{Post}[\delta](s) \cap A \neq \emptyset$ and $\mathit{Post}[\delta](s) \subseteq T$. Since $\mathit{Post}[\delta](s)$ is downward-closed, $\mathit{Post}[\delta](s) \subseteq T$ entails $\mathit{Post}[\delta](s) \subseteq K_\downarrow(T)$. Hence, $s \in \widehat{\mathit{Pre}}_{K_\downarrow(T)}(A)$. The assertion for $\widehat{\mathit{Pre}}^*$ is immediate from the first assertion. $\square$

This allows to rewrite the definition for $\mathit{Safe}(\gamma)$:

$$\begin{aligned}\mathit{Safe}(\gamma) &\stackrel{\text{def}}{=} \nu X.\big(\gamma \sqcap \widehat{\mathit{Pre}}_X(\mathsf{Conf})\big)\\ &= \nu X.\big(\gamma \sqcap \widehat{\mathit{Pre}}_{\downarrow K_\downarrow(X)}(\mathsf{Conf})\big).\end{aligned}$$

One obtains a fixpoint expression which is downward-guarded and hence guarded.

Promising sets: Here too we use Lemma A.1 and rewrite the definition for $Prom(\gamma)$:

$$\begin{aligned} Prom(\gamma) &\stackrel{\text{def}}{=} \nu X.\widehat{Pre}^*_X(\gamma) \\ &= \nu X.\widehat{Pre}^*_{\downarrow K_\downarrow(X)}(\gamma). \end{aligned}$$

One obtains a fixpoint expression which is downward-guarded and hence guarded.

$\exists$CTL properties: Using $Pre(A) = Pre(\uparrow A)$, the definitions for $\exists\big(\gamma \text{ Until } \gamma'\big)$ and $\exists\big(\text{Next}\,\gamma\big)$ (see (*)) can be rewritten in upward-guarded form.

B Proofs for section 4

B.1 Proof of Theorem 4.4

The equivalence of the assertions

1. $\exists\mathcal{U}$ such that $\Pr_{\mathcal{U}}(s \models \Box A) = 1$,
2. $\exists\mathcal{U}$ memoryless such that $\Pr_{\mathcal{U}}(s \models \Box A) = 1$,
3. $s \in Safe(A)$.

follows by the combination of Lemmas B.1 and B.2:

Lemma B.1. *There exists a memoryless scheduler $\mathcal{U}$ such that for all $s \in Safe(A)$,* $\Pr_{\mathcal{U}}(s \models \Box A) = 1$.

Proof. For all $s \in Safe(A)$, there exists a transition rule δ_s such that $Post[\delta_s](s) \subseteq Safe(A)$. In configuration s, $\mathcal{U}$ simply chooses transition δ_s that ensures staying in $Safe(A)$. $\mathcal{U}$ is a memoryless scheduler (it is only based on the current configuration), does not depend on the initial configuration and fulfills $\Pr_{\mathcal{U}}(s \models \Box A) = 1$. $\square$

Lemma B.2. *If* $\Pr_{\mathcal{U}}(s \models \Box A) = 1$ *for some scheduler $\mathcal{U}$, then $s \in Safe(A)$.*

Proof. Assume there exists a scheduler $\mathcal{U}$ such that $\Pr_{\mathcal{U}}(s \models \Box A) = 1$ for some configuration s. Consider the set $T = \{t \in \mathsf{Conf} \mid \Pr_{\mathcal{U}}(s \models \Diamond t) > 0\}$. Then $s \in T$. We show that T is safe for A. Clearly $T \subseteq A$ (otherwise $\Box A$ would not hold almost surely). Let $t \in T$ Because , starting from s, $\mathcal{U}$ may reach t with positive probability, all configurations in $Post[\delta_t](t)$ can be reached too (if δ_t is a rule chosen by $\mathcal{U}$ in t). Thus, $Post[\delta_t](t) \subseteq T$, and T is safe for A. $\square$

B.2 Proof of Theorem 4.7

The equivalence of the following assertions is given by the combination of Lemmas B.3 and B.4 (see below).

1. $\exists\mathcal{U}$ s.t. $\Pr_{\mathcal{U}}(s_0 \models \Diamond A) = 1$,
2. $\exists\mathcal{U}$ memoryless s.t. $\Pr_{\mathcal{U}}(s_0 \models \Diamond A) = 1$,
3. $s_0 \in Prom(A)$.

Lemma B.3. *There exists a memoryless scheduler $\mathcal{U}$ such that for all $s \in \mathit{Prom}(A)$,* $\Pr_{\mathcal{U}}\big(s \models \Diamond A\big) = 1$.

Proof. By definition of $\mathit{Prom}(A)$ one can pick, for each configuration $s \in \mathit{Prom}(A)$, a simple path (that is which never visits twice the same configuration) $\pi_s : s \to \cdots \to A$, witnessing $s \in \mathit{Prom}(A)$. We add the requirement that if some configuration t appears in two different paths π_s and $\pi_{s'}$ then, the suffix should be the same in both paths. These paths are used to build a memoryless scheduler $\mathcal{U}$ that ensures A is eventually visited almost surely. Let us detail $\mathcal{U}$'s behavior. Let t be any configuration of $\mathit{Prom}(A) \smallsetminus A$ (if t is in A, then $\mathcal{U}$ is done). In t, $\mathcal{U}$ chooses the only rule, that appears along the paths. This rule is unique thanks to the requirement we added. The nominal behavior of $\mathcal{U}$ is to follow a path towards a configuration in A but losses can change his plans. Nevertheless, the finite attractor property yields that some configuration $t \in \mathit{Prom}(A)$ will be visited infinitely often and from this configuration, A will be eventually reached using the path π_t. Hence $\Pr_{\mathcal{U}}\big(s \models \Diamond A\big) = 1$ for all $s \in \mathit{Prom}(A)$. □

Lemma B.4. *If* $\Pr_{\mathcal{U}}(s \models \Diamond A) = 1$ *for some scheduler $\mathcal{U}$, then $s \in \mathit{Prom}(A)$.*

Proof. Assume there exists a scheduler $\mathcal{U}$ such that $\Pr_{\mathcal{U}}(s \models \Diamond A) = 1$. Let T be the set of configurations visited by $\mathcal{U}$ before A with positive probability. That is, $T = \{t \in \mathsf{Conf} \,|\, \Pr_{\mathcal{U}}(s \models \neg A \text{ Until } t) > 0\}$. Then s belongs to T. Let us show that T is promising for A. For each $t \in T$ either $t \in A$ (this yields $t \in \mathit{Prom}(A)$) or there exists a $\mathcal{U}$-path π_t of length $m \geq 1$ leading to A. Let π_t be $t = t_0 \xrightarrow{\delta_1} \cdots \xrightarrow{\delta_m} t_m \in A$. Assume $\mathit{Post}[\delta_{i+1}](t_i) \subsetneq \mathit{Prom}(A)$. Then, there exist an index $i \in \{0, \ldots, m-1\}$ and $t'_{i+1} \in \mathit{Post}[\delta_{i+1}](t_i) \cap \mathsf{Conf} \smallsetminus \mathit{Prom}(A)$. Configuration t'_{i+1} is in T, hence $\Pr_{\mathcal{U}}(s \models \neg A \text{ Until } t'_{i+1}) > 0$. And since $t'_{i+1} \notin \mathit{Prom}(A)$, $\Pr_{\mathcal{U}}(t'_{i+1} \models \Diamond A) < 1$. These two inequalities yield $\Pr_{\mathcal{U}}(s \models \Diamond A) < 1$, contradicting the assumption in Lemma B.4. Hence for all indexes i, $\mathit{Post}[\delta_{i+1}](t_i) \subseteq \mathit{Prom}(A)$, *i.e.*, $t \in \mathit{Prom}(A)$. As a consequence $s \in \mathit{Prom}(A)$. □

B.3 Proof of Theorem 4.8.(a)

We have to show the equivalence of the assertions

1. There exists a scheduler $\mathcal{U}$ with $\Pr_{\mathcal{U}}(s \models \Box\Diamond A) = 1$
2. There exists a memoryless scheduler $\mathcal{U}$ with $\Pr_{\mathcal{U}}(s \models \Box\Diamond A) = 1$
3. $s \in \mathit{Prom}^{\geq 1}(A)$.

"(2) $\Longrightarrow$ (1)" is obvious.
"(3) $\Longrightarrow$ (2)": Let $P = \{p \in Q : (p, \varepsilon) \in \mathit{Prom}^{\geq 1}(A)\}$, the projection of $\mathit{Prom}^{\geq 1}(A)$ on Q, the set of locations. In configuration $t = (q, w) \in \mathit{Prom}^{\geq 1}(A)$, $\mathcal{U}$ aims at performing the paths π_p's (given for all $p \in P$): if (q, w) appears on a path π_p, then $\mathcal{U}$ picks the transition rule recommended here, else $\mathcal{U}$ picks the first rule of the path $\pi_q : (q, \varepsilon) \to^+ A$. Because of the finite attractor property, $\mathcal{U}$ reaches eventually A with probability 1, and moreover, visits A infinitely often almost surely. We can choose simple paths (that never visit twice the same configuration) to ensure that $\mathcal{U}$ is memoryless.
"(1) $\Longrightarrow$ (3)": Let $\mathcal{U}$ be a scheduler such that $\Pr_{\mathcal{U}}(s \models \Box\Diamond A) = 1$. Let $T \stackrel{\text{def}}{=} \Big\{t \in \mathsf{Conf} : \Pr_{\mathcal{U}}\big(s \models \Diamond t\big) > 0\Big\}$. We show that T is a subset of $\mathit{Prom}^{\geq 1}(A)$. Let $t \in T$. Then, there is

a $\mathcal{U}$-path of the form $s \xrightarrow{+} t$. This $\mathcal{U}$-path can be extended to an infinite $\mathcal{U}$-path where $\Box\Diamond A$ holds. Thus, there is a fragment of a $\mathcal{U}$-path of the form $t \xrightarrow{+} A$, and by definition of T, all possible configurations along this path are in T. Hence T satisfies $\forall t \in T$, $\exists t = t_0 \xrightarrow{\delta_1} t_1 \cdots \xrightarrow{\delta_m} t_m$ with $m \geq 1$, $t_m \in A$ and $Post[\delta_i](t_{i-1}) \subseteq T$. Because $Prom^{\geq 1}(A)$ is the largest set with this property, we get $T \subseteq Prom^{\geq 1}(A)$. $\Box$

B.4 Proof of Theorem 4.8.(d)

We have to show the equivalence of the assertions:

1. There exists a scheduler $\mathcal{U}$ with $\Pr_{\mathcal{U}}(s \models \Diamond\Box A) > 0$.
2. There exists a memoryless scheduler $\mathcal{U}$ with $\Pr_{\mathcal{U}}(s \models \Diamond\Box A) > 0$.
3. $s \in Pre^*(Safe(A))$

"(2) $\Longrightarrow$ (1)" is obvious.
"(3) $\Longrightarrow$ (2)" : Assume s can reach $Safe(A)$ via a shortest path $s \xrightarrow{*} Safe(A)$. Scheduler $\mathcal{U}$ will first try to follow this path, and reaches $Safe(A)$ with positive probability. When/if in $Safe(A)$, $\mathcal{U}$ behaves as the memoryless scheduler describe in Lemma B.2, that ensures staying in A almost surely.
"(1) $\Longrightarrow$ (3)": Let T be the set of configurations t such that $\Pr_{\mathcal{U}}(s \models \Box\Diamond t \wedge \Diamond\Box A) > 0$. Then, $T \subseteq A$ and $\Pr_{\mathcal{U}}(s \models \Diamond T) > 0$. We show that T is safe for A. This will end the proof since $s \xrightarrow{*} T$.

For any $t \in T$ there is a transition rule δ_t that is enabled in t such that $\Pr_{\mathcal{U}}(s \models \Box\Diamond t \wedge \Box\Diamond$"$\delta_t$ is chosen in t"$\wedge \Diamond\Box A) > 0$. But then also

$$\Pr_{\mathcal{U}}(s \models \Box\Diamond t \wedge \Box\Diamond\text{"}\delta_t \text{ is chosen in } t\text{"} \wedge \bigwedge_{t' \in Post[\delta_t](t)} \Box\Diamond t \wedge \Diamond\Box A) > 0.$$

Hence, $Post[\delta_t](t) \subseteq T$. This yields $T \subseteq Safe(A)$. $\Box$

B.5 Proof of Theorem 4.8.(b)

We consider the set $T_{\Box\Diamond A}$ of all configurations $t \in \mathsf{Conf}$ such that $\Pr_{\mathcal{V}}(t \models \Box\Diamond A) = 1$ for some (memoryless) scheduler $\mathcal{V}$. Then, $T_{\Box\Diamond A} = Prom^{\geq 1}(A)$ (by (a)) and the following assertions are equivalent:

1. $\Pr_{\mathcal{U}}(s \models \Box\Diamond A) > 0$ for some finite-memory scheduler $\mathcal{U}$.
2. $\Pr_{\mathcal{U}}(s \models \Box\Diamond A) > 0$ for some memoryless scheduler $\mathcal{U}$.
3. $s \in Pre^*(T_{\Box\Diamond A})$.

"(1) $\Longrightarrow$ (2)": obvious.
"(3) $\Longrightarrow$ (2)": Let $\mathcal{U}$ be a memoryless scheduler which first generates a shortest path from s to F with positive probability. As soon as $\mathcal{U}$ reaches $T_{\Box\Diamond A}$ (this happens with positive probability), $\mathcal{U}$ behaves as the memoryless scheduler such that $\Pr_{\mathcal{V}}(t \models \Box\Diamond A) = 1$ for all $t \in T_{\Box\Diamond A}$. $\mathcal{U}$ is memoryless and satisfies the desired property.
"(1) $\Longrightarrow$ (3)": The finite attractor property yields the existence of some $t \in \mathsf{Conf}$ such that

$$\Pr_{\mathcal{U}}(s \models \Box\Diamond t \wedge \Box\Diamond A) > 0.$$

Clearly, we then have $s \to^* t$. Thus, it suffices to show that $t \in T_{\Box\Diamond A}$.

Let T be the set of all configurations that are reachable from t in the Markov chain for $\mathcal{U}$.

$$\Pr_{\mathcal{U}}\left(s \models \Box\Diamond t \wedge \wedge\Diamond\Box T \wedge \Box\Diamond A\right) > 0.$$

This yields $T \cap A \neq \emptyset$. If t is visited infinitely often then almost surely all configurations $t' \in T$ are visited infinitely often too, because $\mathcal{U}$ is memoryless. That is

$$\Pr_{\mathcal{U}}\left(t \models \bigwedge_{t' \in T} \Box\Diamond t' \wedge \Box\Diamond A\right) > 0.$$

Hence $\Pr_{\mathcal{U}}(t \models \Box\Diamond A) = 1$. Thus, $t \in T_{\Box\Diamond A}$. □

B.6 Proof of Theorem 4.8.(c)

we show the equivalence of the assertions:

1. There exists a scheduler $\mathcal{U}$ such that $\Pr_{\mathcal{U}}(s \models \Diamond\Box A) = 1$.
2. $\Pr_{\mathcal{U}}(s \models \Diamond\Box A) = 1$ for some memoryless scheduler $\mathcal{U}$.
3. $s \in \mathit{Prom}(\mathit{Safe}(A))$.

"(2) $\Longrightarrow$ (1)": obvious.
"(3) $\Longrightarrow$ (2)": $\mathcal{U}$ uses $\mathcal{W}$ to reach $\mathit{Safe}(A)$ with probability one, and then mimics $\mathcal{V}$ to ensure that $\Box A$ holds (almost) surely. $\mathcal{U}$ is memoryless, as safe and stubborn schedulers are.
"(1) $\Longrightarrow$ (3)": Let $\Pr_{\mathcal{U}}(s \models \Diamond\Box A) = 1$ and consider $T = \{t \in \mathsf{Conf} : \Pr_{\mathcal{U}}(s \models \Box\Diamond t) > 0\}$. By the attractor property, we have $\Pr_{\mathcal{U}}(s \models \Diamond T) = 1$. We now show that $T \subseteq \mathit{Safe}(A)$. Observe first that $T \subseteq A$ as we have $\Pr_{\mathcal{U}}(s \models \Box\Diamond t \wedge \Diamond\Box A) > 0$ for all $t \in T$. For any $t \in T$ there is a transition rule δ_t which is enabled in t and

$$\Pr_{\mathcal{U}}\left(s \models \Box\Diamond t \wedge \Box\Diamond\text{"}\delta_t \text{ is chosen for } t\text{"}\right) > 0.$$

Almost surely, if t is visited infinitely often and δ_t taken infinitely often in t then all δ_t-successors of t are visited infinitely often. This yields

$$\Pr_{\mathcal{U}}\left(s \models \Box\Diamond t \wedge \Box\Diamond\text{"}\delta_t \text{ is chosen for } t\text{"} \wedge \bigwedge_{t' \in \mathit{Post}[\delta_t](t)} \Box\Diamond t'\right) > 0.$$

This yields $\mathit{Post}[\delta_t](t) \subseteq T$. Thus, $T \subseteq \mathit{Safe}(A)$ and $s \in \mathit{Prom}(\mathit{Safe}(A))$. □

B.7 Proof of Lemma 4.9

Let $T = T_{\Box\Diamond A \wedge \Box\neg B}$.
"(2) $\Longrightarrow$ (1)": obvious.
"(3) $\Longrightarrow$ (2)": Assume $s \to^* T$. We build a memoryless scheduler $\mathcal{U}$ that achieves $\Pr_{\mathcal{U}}(s \models \Box\Diamond A \wedge \Diamond\Box\neg B) > 0$. $\mathcal{U}$ first tries to reach T using the path $s \to^* T$. It succeeds with positive probability. When in T, $\mathcal{U}$ behaves as the memoryless scheduler associated with $\mathit{Prom}^{\geq 1}_{\neg B}(A)$. $\mathcal{U}$ is memoryless if we pick a shortest path from s to T.
"(1) $\Longrightarrow$ (3)": Let $\mathcal{U}$ be a finite-memory scheduler such that $\Pr_{\mathcal{U}}(s \models \Box\Diamond A \wedge \Diamond\Box\neg B) > 0$. Let p be any control state such that $\Pr_{\mathcal{U}}(s \models \Box\Diamond(p,\varepsilon) \wedge \Box\Diamond A \wedge \Diamond\Box\neg B)) > 0$. Such a location p exists thanks to the finite attractor property. Moreover $t \in T$, and $s \to^* t$. Hence $s \to^* T$. □

B.8 Proof of Theorem 4.10

The equivalence of (2) and (3) is due to Theorem 4.7.

"(2) $\Longrightarrow$ (1)": $\mathcal{U}$ first aims at reaching T_S, using the promising scheduler $\mathcal{V}$. When some T_i is reached $\mathcal{U}$ can stay in T_i and achieves $\Pr_{\mathcal{U}}\big(s \models \Box\Diamond A_i \wedge \Diamond\Box\neg B_i\big) = 1$. If $\mathcal{V}$ is finite-memory, then $\mathcal{U}$ is finite-memory too.

"(1) $\Longrightarrow$ (2)": Assume $\Pr_{\mathcal{U}}\big(s \models \bigvee_{i=1}^{n} \Box\Diamond A_i \wedge \Diamond\Box\neg B_i\big) = 1$ for some finite-memory scheduler $\mathcal{U}$. Let T be the set of all configurations $t \in \mathsf{Conf}$ such that $\Pr_{\mathcal{U}}\big(s \models \Box\Diamond t \wedge \neg\varphi_S\big) >\big)$. As a consequence of the attractor property, T is non-empty and $\Pr_{\mathcal{U}}(s \models \Diamond T) = 1$. Thus, $s \in \mathit{Prom}(T)$. Let us show that $T \subseteq T_S$. From this, we obtain $\mathit{Prom}(T) \subseteq \mathit{Prom}(T_S)$ by the monotonicity of $\mathit{Prom}(\cdot)$).

Let $t \in T$. Then, there exists some index i such that $\Pr_{\mathcal{U}}(s \models \Box\Diamond t \wedge \Box\Diamond A_i \wedge \Diamond\Box\neg B_i) > 0$. Since $\mathcal{U}$ is memoryless, this implies: $\Pr_{\mathcal{U}}(s \models \Box\Diamond t \wedge \Box\Diamond t' \wedge \Box\Diamond A_i \wedge \Diamond\Box\neg B_i) > 0$ for all configuration t' which is a $\mathcal{U}$-successor of t. Consequently, all the $\mathcal{U}$-successors of t are in $\neg B_i$, and $t \xrightarrow{*} A_i$. This yields $\Pr_{\mathcal{U}}(t \models \Box\Diamond A_i \wedge \Diamond\Box\neg B_i) = 1$, that is $t \in T_i$. Hence, $T \subseteq \bigcup T_i = T_S$ and $s \in \mathit{Prom}(T_S)$. $\Box$

C Proofs for section 5

Lemma C.1. *For each $\mathcal{F} \subseteq 2^{\Delta}$, there exists a $\mathcal{F}$-fair fm-scheduler.*

Proof. We describe a finite-memory scheduler $\mathcal{U}$ whose modes are the permutations $(f_1,\dots,f_n)$ of the elements of $\mathcal{F}$. The starting mode is arbitrary. In mode $(f_1,\dots,f_n)$ and for the current configuration s, the decision by $\mathcal{U}$ is as follows. Let $F = \{f \in \mathcal{F} : f \cap \Delta(s) \neq \emptyset\}$ be the set of fairness sets that are enabled in s. If $F = \emptyset$ then $\mathcal{U}$ selects an arbitrary transition rule $\delta_s \in \Delta(s)$ and stays in mode $(f_1,\dots,f_n)$. If $F \neq \emptyset$ and $i = \min\{j : f_j \cap \Delta(s) \neq \emptyset\}$ then $\mathcal{U}$ selects a transition rule $\delta_{s,i} \in f_i \cap \Delta(s)$ and switches to the mode $(f_1,\dots,f_{i-1},f_{i+1},\dots,f_n,f_i)$. In this way, we obtain a fm-scheduler $\mathcal{U}$ where all infinite $\mathcal{U}$-paths are $\mathcal{F}$-fair. In particular, $\mathcal{U}$ is $\mathcal{F}$-fair. $\Box$

Theorem C.2. *(a) There exists a $\mathcal{F}$-fair scheduler $\mathcal{V}$ such that* $\Pr_{\mathcal{V}}(s \models \Diamond A) > 0$ *if and only if there exists a scheduler $\mathcal{U}$ such that* $\Pr_{\mathcal{U}}(s \models \Diamond A) > 0$.
(b) There exists a $\mathcal{F}$-fair scheduler $\mathcal{V}$ such that $\Pr_{\mathcal{V}}(s \models \Diamond A) = 1$ *if and only if there exists a scheduler $\mathcal{U}$ such that* $\Pr_{\mathcal{U}}(s \models \Diamond A) = 1$.

Proof. In (a) and (b) the "only if"-part is trivial. Let us assume we have a scheduler $\mathcal{U}$ with $\Pr_{\mathcal{U}}(s \models \Diamond A) > 0$. We pick an arbitrary shortest $\mathcal{U}$-path π from s to some configuration $t \in A$. Scheduler $\mathcal{V}$ behaves as $\mathcal{U}$ for all proper prefixes of π. As soon as t is reached, as well as for all paths that are not prefixes of π, $\mathcal{V}$ behaves $\mathcal{F}$-fair (see Lemma C.1). Thus, $\mathcal{V}$ is a $\mathcal{F}$-fair scheduler with $\Pr_{\mathcal{V}}(s \models \Diamond A) \geq \Pr(\pi) > 0$.

Let us now assume that $\mathcal{U}$ is a scheduler with $\Pr_{\mathcal{U}}(s \models \Diamond A) = 1$. Let $\mathcal{V}$ be a scheduler that behaves as $\mathcal{U}$ for all paths π that do not contain an A-configuration. As soon as A is reached, $\mathcal{V}$ behaves $\mathcal{F}$-fair (see Lemma C.1). Since $\Pr_{\mathcal{U}}(s \models \Diamond A) = 1$, $\mathcal{V}$ is $\mathcal{F}$-fair and $\Pr_{\mathcal{V}}(s \models \Diamond A) = 1$. $\Box$

Lemma C.3. *There exists a $\mathcal{F}$-fair finite-memory scheduler $\mathcal{V}$ such that* $\Pr_{\mathcal{V}}(t \models \Box A) = 1$ *for all $t \in \mathit{Safe}_{\mathcal{F}}(A)$.*

Proof. We first show that for each $F \subseteq \mathcal{F}$ there exists a finite-memory $\mathcal{F}$-fair scheduler $\mathcal{V}_F$ such that $\Pr_{\mathcal{V}_F}(t \models \Box A) = 1$ for all $t \in \mathit{Safe}[F](A)$. This is obvious for $F = \emptyset$ where even a memoryless scheduler $\mathcal{V}_\emptyset$ with $\Pr_{\mathcal{V}_\emptyset}(t \models \Box A) = 1$ for all $t \in \mathit{Safe}_{\mathcal{F}}[\emptyset](A) = \mathit{Safe}(A \setminus \mathsf{Enabl}(\mathcal{F}))$ exists.

We now regard a nonempty subset F of $\mathcal{F}$, say $F = \{f_0, \ldots, f_{k-1}\}$. For each state $t \in \mathit{Safe}[F](A)$ and each $f \in F$ we fix a shortest path $\pi_{t,f}$ of the form

$$\pi_{t,f} \; = \; s_0 \xrightarrow{\delta_1} \ldots\ldots \xrightarrow{\delta_m} s_m$$

as in the definition of $\mathit{Safe}[F](A)$, i.e., $s_0 = t$, $m \geq 1$, $\delta_m \in f$ and $\delta_i \notin f \wedge \mathit{Post}[\delta_i](s_{i-1}) \subseteq \mathit{Safe}[F](A)$ for all $1 \leq i \leq m$. We may assume that $\pi_{s_i,f}$ agrees with the suffix of $\pi_{t,f}$ starting in s_i. We design a finite-memory scheduler $\mathcal{V}_F$ that works with the modes $f \in F$. In mode f and current configuration $t \in \mathit{Safe}[F](A)$, $\mathcal{V}_F$ attempts to generate the path $\pi_{t,f}$, i.e., it chooses the first transition rule δ_1 of $\pi_{t,f}$ in configuration t and mode f. Since $\mathit{Post}[\delta_1](t) \subseteq \mathit{Safe}[F](A)$ all configurations visited by $\mathcal{V}$ belong to $\mathit{Safe}[F](A)$. As soon as a transition rule in f has been taken (the finite attractor property ensures that this will happen with probability 1), $\mathcal{V}_F$ switches from mode $f = f_j$ to mode f' where $f' = f_{(j+1) \bmod k}$. The $\mathcal{V}_F$-paths are almost surely $\mathcal{F}$-fair as, with probability 1, for each $f \in F$ infinitely many f-transition rules are taken and the other fairness sets $g \in \mathcal{F} \setminus F$ are not enabled in the configurations of $\mathit{Safe}[F](A)$. Thus, $\mathcal{V}_F$ is finite-memory and $\mathcal{F}$-fair and – since $\mathit{Safe}[F](A) \subseteq A$ – fulfills $\Pr_{\mathcal{V}_F}(t \models \Box A) = 1$ for all $t \in \mathit{Safe}[F](A)$.

It remains to compose schedulers $\mathcal{V}_F$ for $F \subseteq \mathcal{F}$ to obtain a $\mathcal{F}$-fair finite-memory scheduler $\mathcal{V}$ with $\Pr_{\mathcal{V}}(t \models \Box A) = 1$ for all $t \in \mathit{Safe}_{\mathcal{F}}(A)$. Let $F_1, \ldots, F_m$ be an enumeration of the subsets of $\mathcal{F}$ (hence, $m = 2^{|\mathcal{F}|}$) such that $F_i \subseteq F_j$ implies $i \leq j$ (hence, $F_1 = \emptyset$ and $F_m = \mathcal{F}$). Furthermore, let $T_1 = \mathit{Safe}_{\mathcal{F}}[F_1](A)$ and $T_i = \mathit{Safe}_{\mathcal{F}}[F_i](A) \setminus (T_1 \cup \cdots \cup T_{i-1})$ for $1 < i \leq m$. Each configuration $t \in \mathit{Safe}_{\mathcal{F}}(A)$ belongs to exactly one T_i. For $t \in T_i$ and $f \in F_i$ let path $\pi_{t,f}$ be as above. Then,

$$\mathit{Post}[\delta_i](s_{i-1}) \in T_1 \cup \cdots \cup T_{i-1} \cup T_i \text{ for all } 1 \leq i \leq m.$$

Let $\mathcal{V}$ be a finite-memory scheduler with modes (i, f) where $1 \leq i \leq m$ and $f \in F_i$. For the starting configuration t, $\mathcal{V}$ starts in mode i provided that $t \in T_i$. In mode (i, f) and configuration t, $\mathcal{V}$ behaves as $\mathcal{V}_{F_i}$ in mode f for t , i.e., $\mathcal{V}$ attempts to generate the path $\pi_{t,f}$. As long as the current configuration belongs to T_i, $\mathcal{V}$ continues to simulate $\mathcal{V}_{F_i}$. As soon as a configuration $t' \in T_j$ for some $j < i$ has been reached, $\mathcal{V}$ switches to mode (j, f') for some arbitrary $f' \in F_j$ and behaves as $\mathcal{V}_{F_j}$. The so obtained scheduler $\mathcal{V}$ is finite-memory, $\mathcal{F}$-fair and fulfills $\Pr_{\mathcal{V}}(t \models \Box A) = 1$ for all $t \in \mathit{Safe}_{\mathcal{F}}(A)$. ☐

Before presenting the proof of Theorem 5.3 we recall the definition of strong prob-fairness and state some essential properties.

Definition C.4 ((Strong) prob-fairness). *Let* π *be an infinite path.* $\inf(\pi)$ *denotes the set of configurations that appear infinitely often in* π. π *is called*

- *prob-fair if* $\inf(\pi) \neq \emptyset$ *and for all* $s \in \inf(\pi)$ *and all transition rules* $\delta \in \Delta(s)$ *that are taken infinitely often in* π, *all steps* $s \xrightarrow{\delta} t$ *appear infinitely often in* π,
- *strongly prob-fair if* π *is prob-fair and there exists an (infinite) suffix* π' *of* π *such that each finite path fragment in* π' *appears infinitely often in* π' *(and* π*).*

For fm-scheduler $\mathcal{V}$, almost all $\mathcal{V}$-paths are strongly prob-fair.

Lemma C.5. *If* π *is a* $\mathcal{F}$ *-fair and strongly prob-fair path with* $\pi \models \Box A$ *then* $\inf(\pi) \subseteq$ $Safe_{\mathcal{F}}(A)$.

Proof. Since π is $\mathcal{F}$-fair there exists $F \subseteq \mathcal{F}$ such that each $f \in \mathcal{F} \setminus F$ is enabled in π only finitely many times, while for each $f \in F$ there is some transition rule $\delta \in f$ which is taken infinitely often in π. We show that $\inf(\pi)$ satisfies the conditions in the definition of $Safe[F](A)$. Since $Safe[F](A)$ is defined as the largest set of configurations where these conditions hold we obtain $\inf(\pi) \subseteq Safe[F](A)$.

For the special case $F = \emptyset$ we have $Safe[\emptyset](A) = Safe(A \setminus \mathsf{Enabl}(\mathcal{F}))$. By definition of F and since $\pi \models \Box A$ we have $\inf(\pi) \subseteq A \subseteq A \setminus \mathsf{Enabl}(\mathcal{F}))$. Moreover, if $t \in \inf(\pi)$ and $\delta \in \Delta(t)$ is a transition rule that is taken infinitely often for t in π then $Post[\delta](t) \subseteq \inf(\pi)$ since π is prob-fair.

Let us now assume that F is nonempty. The first condition requires $\inf(\pi) \subseteq A \setminus \mathsf{Enabl}(\mathcal{F} \setminus F)$. This is clear by the definition of F and since $\pi \models \Box A$. Let $t \in \inf(\pi)$ and $f \in F$. Since π is strongly prob-fair, there exists a finite path

$$t = s_0 \xrightarrow{\delta_1} s_1 \xrightarrow{\delta_2} \cdots \xrightarrow{\delta_m} s_m \text{ with } \delta_m \in f$$

that appears infinitely often in π. Since $\pi \models \Box A$ and π is prob-fair we have $Post[\delta_i](s_{i-1}) \subseteq \inf(\pi)$ for all $1 \leq i \leq m$. □

Proof (of thm:fair-pos-prob-invariant). We first observe that there is a fm-scheduler $\mathcal{V}$ such that $\Pr_{\mathcal{V}}(t \models \Box A) = 1$ for all states $t \in Safe_{\mathcal{F}}(A)$. (For the construction of $\mathcal{V}$ see Lemma C.3.)

If π is an infinite path then let $\inf(\pi)$ denote the set of configurations that appear infinitely often in π. We call π *strongly prob-fair* iff $\inf(\pi) \neq \emptyset$ and π has an infinite suffix π' such that each finite path fragment of π' appears infinitely often in π' (and π). The finite attractor property yields that for each fm-scheduler $\mathcal{V}$, almost all paths are strongly prob-fair. We now observe:

$$\inf(\pi) \subseteq Safe_{\mathcal{F}}(A) \text{ if } \pi \text{ is strongly prob-fair and } \mathcal{F}\text{-fair path with } \pi \models \Box A \qquad (*)$$

Equation (*) is given by Lemma C.5.

ad (a). Let $\mathcal{V}$ be a $\mathcal{F}$-fair fm-scheduler with $\Pr_{\mathcal{V}}(s \models \Box A) > 0$. Let π be an infinite $\mathcal{V}$-path where $\Box A$ holds and which is $\mathcal{F}$-fair and strongly prob-fair. Since almost all $\mathcal{V}$-paths are $\mathcal{F}$-fair and strongly prob-fair, such a path π exists. Statement (*) yields $\inf(\pi) \subseteq Safe_{\mathcal{F}}(A)$, and thus, $s \models \exists(A \text{ Until } Safe_{\mathcal{F}}(A))$.

Let us now assume that $s \models \exists(A \text{ Until } Safe_{\mathcal{F}}(A))$. Then, there is some $F \subseteq \mathcal{F}$ such that $s \models \exists(A \text{ Until } Safe[F](A))$. Let $s \xrightarrow{\delta_1} s_1 \xrightarrow{\delta_2} \cdots \xrightarrow{\delta_m} s_m$ be a path from s to some configuration $s_m \in Safe[F](A)$ such that $s, s_1, \ldots, s_{m-1} \in A$. Let $\mathcal{V}$ be a scheduler that attempts to generate this path. If $\mathcal{V}$ fails then $\mathcal{V}$ behaves in an arbitrary, but $\mathcal{F}$-fair way. Otherwise when $Safe[F](A)$ is reached via the selected path from s to s_m, $\mathcal{V}$ behaves as $\mathcal{W}$, where $\mathcal{W}$ is a $\mathcal{F}$-fair fm-scheduler with $\Pr_{\mathcal{W}}(t \models \Box A) = 1$ for all $t \in Safe_{\mathcal{F}}(A)$. Clearly, $\mathcal{V}$ is finite-memory and $\mathcal{F}$-fair and fulfills $\Pr_{\mathcal{V}}(s \models \Box A) > 0$.

ad (b). We first observe that $s \in Prom_A(C)$ iff $s \models \Pr_{\mathcal{U}}(s \models A \text{ Until } C) = 1$ for some memoryless scheduler $\mathcal{U}$. (The proof is similar to the one of Theorem 4.7.)

Let $\mathcal{V}$ be a $\mathcal{F}$-fair fm-scheduler such that $\Pr_{\mathcal{V}}(s \models \Box A) = 1$. Let $T = \bigcup_\pi \inf(\pi)$ where π ranges over all $\mathcal{F}$-fair and strongly prob-fair $\mathcal{V}$-path π with $\pi \models \Box A$. By (*) we get: $T \subseteq \mathit{Safe}_{\mathcal{F}}(A)$. Since $\mathcal{V}$ is finite-memory, the $\mathcal{F}$-fair and strongly prob-fair $\mathcal{V}$-paths have probability measure 1. This yields $\Pr_{\mathcal{U}}(s \models A \text{ Until } \mathit{Safe}_{\mathcal{F}}(A)) = \Pr_{\mathcal{U}}(s \models A \text{ Until } T) = 1$. Hence, $s \in \mathit{Prom}_A(\mathit{Safe}_{\mathcal{F}}(A))$.

Let now $s \in \mathit{Prom}_A(C)$ where $C = \mathit{Safe}_{\mathcal{F}}(A)$. Then, $\Pr_{\mathcal{U}}(s \models (A \text{ Until } C)) = 1$ for some memoryless scheduler $\mathcal{U}$. We now modify $\mathcal{U}$ to obtain a $\mathcal{F}$-fair fm-scheduler $\mathcal{V}$ with $\Pr_{\mathcal{V}}(s \models \Box A) = 1$. $\mathcal{V}$ mimics $\mathcal{U}$ as long as C has not yet been reached. Having reached C, $\mathcal{V}$ behaves as a $\mathcal{F}$-fair fm-scheduler $\mathcal{W}$ with $\Pr_{\mathcal{W}}(t \models \Box A) = 1$ for all $t \in \mathit{Safe}_{\mathcal{F}}(A)$. Hence $\mathcal{V}$ is finite-memory, $\mathcal{F}$-fair, and fulfills $\Pr_{\mathcal{V}}(s \models \Box A) = 1$. $\Box$

Theorem C.6 (cf. Theorem 5.3). *Let $A \subseteq \mathsf{Conf}$ and $s \in \mathsf{Conf}$. Then:*

(a) There exists a $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$ with $\Pr_{\mathcal{U}}(s \models \Box\Diamond A) = 1$ iff $s \in \mathit{Prom}(T^{\mathcal{F}}_{\Box\Diamond A})$.
(b) There exists a $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$ with $\Pr_{\mathcal{U}}(s \models \Box\Diamond A) = 1$ iff $s \in \mathit{Pre}^*(T^{\mathcal{F}}_{\Box\Diamond A})$.
(c) There exists a $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$ with $\Pr_{\mathcal{U}}(s \models \Diamond\Box A) = 1$ iff $s \in \mathit{Prom}(T^{\mathcal{F}}_{\Box A})$.
(d) There exists a $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$ with $\Pr_{\mathcal{U}}(s \models \Diamond\Box A) > 0$ iff $s \in \mathit{Pre}^*(T^{\mathcal{F}}_{\Box A})$.

Proof. The proofs for assertions (a)-(d) rely on the following observations:

(1) There exists a $\mathcal{F}$-fair fm-scheduler $\mathcal{W}$ with $\Pr_{\mathcal{W}}(t \models \Box\Diamond A) = 1$ for all $t \in T^{\mathcal{F}}_{\Box\Diamond A}$.
(2) If π is a strongly prob-fair and $\mathcal{F}$-fair path with $\pi \models \Box\Diamond A$ then $\inf(\pi) \subseteq T^{\mathcal{F}}_{\Box\Diamond A}$.
(3) There exists a $\mathcal{F}$-fair fm-scheduler $\mathcal{W}$ with $\Pr_{\mathcal{W}}(t \models \Diamond A) = 1$ for all $t \in T^{\mathcal{F}}_{\Box A}$.
(4) If π is a strongly prob-fair and $\mathcal{F}$-fair path with $\pi \models \Diamond\Box A$ then $\inf(\pi) \subseteq T^{\mathcal{F}}_{\Box A}$.

(1) and (3) can be shown with similar arguments as in Lemma C.3. The proof of (2) and (4) is similar to Lemma C.5.

ad (a). Let $s \in \mathit{Prom}(T^{\mathcal{F}}_{\Box\Diamond A})$. Then, $\Pr_{\mathcal{V}}(s \models \Diamond T^{\mathcal{F}}_{\Box\Diamond A}) = 1$ for some memoryless scheduler $\mathcal{V}$ (Theorem 4.7). Let $\mathcal{W}$ be a $\mathcal{F}$-fair fm-scheduler $\mathcal{W}$ such that $\Pr_{\mathcal{W}}(t \models \Box\Diamond A) = 1$ for all configurations $t \in T^{\mathcal{F}}_{\Box\Diamond A}$ (see (1)). Schedulers $\mathcal{V}$ and $\mathcal{W}$ can be combined to obtain a $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$ with $\Pr_{\mathcal{U}}(s \models \Box\Diamond A) = 1$.

Let us now assume that $\Pr_{\mathcal{U}}(s \models \Box\Diamond A) = 1$ for some $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$. Then, almost all $\mathcal{U}$-paths π are strongly prob-fair and $\mathcal{F}$-fair and fulfill $\Box\Diamond A$. Since $\inf(\pi) \subseteq T^{\mathcal{F}}_{\Box\Diamond A})$ for each such path π (see (2)), we obtain $\Pr_{\mathcal{U}}(s \models \Diamond T^{\mathcal{F}}_{\Box\Diamond A}) = 1$. This implies $s \in \mathit{Prom}(T^{\mathcal{F}}_{\Box\Diamond A})$ by Theorem 4.7.

ad (b). Let $s \in \mathit{Pre}^*(T^{\mathcal{F}}_{\Box\Diamond A})$. We fix a finite shortest path π from s to some configuration $t \in T^{\mathcal{F}}_{\Box\Diamond A}$. We design a $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$ as follows. Scheduler $\mathcal{U}$ first attempts to generate the path π. If it fails, $\mathcal{U}$ continues in an arbitrary but $\mathcal{F}$-way way. If state t has been reached via π then $\mathcal{V}$ simulates a fixed $\mathcal{F}$-fair fm scheduler $\mathcal{V}$ with $\Pr_{\mathcal{V}}(u \models \Box\Diamond A) = 1$ for all $u \in T^{\mathcal{F}}_{\Box\Diamond A}$. Then, $\mathcal{U}$ is in fact $\mathcal{F}$-fair and finite-memory and we have $\Pr_{\mathcal{U}}(s \models \Box\Diamond A) \geq \Pr(\pi) > 0$.

Let us now assume that $\mathcal{U}$ is a $\mathcal{F}$-fair finite-memory scheduler with $\Pr_{\mathcal{U}}(s \models \Box\Diamond A) > 0$. Let π be a strongly prob-fair and $\mathcal{F}$-fair $\mathcal{U}$-path with $\pi \models \Box\Diamond A$. Then, $\inf(\pi) \subseteq T^{\mathcal{F}}_{\Box\Diamond A}$ by (2). Thus, $s \in \mathit{Pre}^*(T^{\mathcal{F}}_{\Box\Diamond A})$.

ad (c). (c) relies on the equivalence of the following assertions:

1 $\exists \mathcal{U}$ $\mathcal{F}$-fair s.t. $\Pr_{\mathcal{U}}(s \models \Diamond\Box A) = 1$
2 $\exists \mathcal{V}$ $\mathcal{F}$-fair s.t. $\Pr_{\mathcal{V}}(s \models \Diamond T^{\mathcal{F}}_{\Box A}) = 1$
3 $\exists \mathcal{W}$ s.t. $\Pr_{\mathcal{W}}(s \models \Diamond T^{\mathcal{F}}_{\Box A}) = 1$, i.e., $s \in \mathit{Prom}(T^{\mathcal{F}}_{\Box A})$ (by Theorem 4.7).

"2 $\Longleftrightarrow$ 3:" follows from part (b) of Theorem C.2.
"1 $\Longrightarrow$ 2": Let π be an infinite $\mathcal{U}$-path starting in s that is strongly prob-fair and $\mathcal{F}$-fair and where $\Diamond\Box A$ holds. Then, $\inf(\pi) \subseteq T^{\mathcal{F}}_{\Box A}$ by (4). Since the $\mathcal{F}$-fair and prob-fair $\mathcal{U}$-paths where $\Diamond\Box A$ holds have probability measure 1, we get $\Pr_{\mathcal{U}}(s \models \Diamond T^{\mathcal{F}}_{\Box A}) = 1$.
"3 $\Longrightarrow$ 1": Let $\Pr_{\mathcal{W}}(s \models \Diamond T^{\mathcal{F}}_{\Box A}) = 1$ for some memoryless scheduler $\mathcal{W}$ (cf. Theorem 4.7). Furthermore, there is a $\mathcal{F}$-fair fm-scheduler $\mathcal{W}'$ with $\Pr_{\mathcal{W}'}(t \models \Box A) = 1$ for all $t \in T^{\mathcal{F}}_{\Box A}$ (see (3)). Composing $\mathcal{W}$ and $\mathcal{W}'$ yields a $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$ with $\Pr_{\mathcal{U}}(s \models \Diamond\Box A) = 1$.

ad (d). If $\mathcal{U}$ is a $\mathcal{F}$-fair fm-scheduler with $\Pr_{\mathcal{U}}(s \models \Diamond\Box A) > 0$ then the probability measure of the prob-fair and $\mathcal{F}$-fair $\mathcal{U}$-paths π where $\Diamond\Box A$ holds is positive. For each such path π, we have $\inf(\pi) \subseteq T^{\mathcal{F}}_{\Box A}$ by (4). This yields $s \in \mathit{Pre}^*(T^{\mathcal{F}}_{\Box A})$.

Vice versa, let $s \in \mathit{Pre}^*(T^{\mathcal{F}}_{\Box A})$. We fix an arbitrary shortest path π from s to some $t \in T^{\mathcal{F}}_{\Box A}$. Then, we consider a finite-memory $\mathcal{F}$-fair scheduler $\mathcal{U}$ which, when started in s, attempts to generate π. If it fails then $\mathcal{U}$ behaves in an arbitrary, but $\mathcal{F}$-fair way (Lemma C.1). After having reached t, scheduler $\mathcal{U}$ behaves as a finite-memory $\mathcal{F}$-fair scheduler $\mathcal{V}$ where $\Pr_{\mathcal{V}}(t' \models \Box A) = 1$ for all $t' \in T^{\mathcal{F}}_{\Box A}$. We then have $\Pr_{\mathcal{U}}(s \models \Diamond\Box A) > 0$. $\Box$

Lemma C.7 (Parts (c) and (d) of Lemma 5.4). *Let s be a configuration in $\mathcal{N}$ (and $\mathcal{N}'$) and φ an LTL formula. Then:*

(a) There exists a $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$ for $\mathcal{N}$ such that $\Pr_{\mathcal{U}}(s \models \varphi) > 0$ *iff there exists a fm-scheduler $\mathcal{V}$ for $\mathcal{N}'$ such that* $\Pr_{\mathcal{V}}(s \models \mathit{fair} \wedge \varphi') > 0$.
(b) There exists a $\mathcal{F}$-fair fm-scheduler $\mathcal{U}$ for $\mathcal{N}$ such that $\Pr_{\mathcal{U}}(s \models \varphi) < 1$ *iff there exists a fm-scheduler $\mathcal{V}$ for $\mathcal{N}'$ such that* $\Pr_{\mathcal{V}}(s \models \mathit{fair} \wedge \neg\varphi') > 0$.

Proof. (b) follows from (a) using the fact that $\Pr_{\mathcal{U}}(s \models \varphi) = 1 - \Pr_{\mathcal{U}}(s \models \neg\varphi)$. We now provide the proof for (a).

In the sequel, we use the following notation. If $\pi_{\mathcal{N}}$ is a finite path in $\mathcal{N}$ then $\pi_{\mathcal{N}'}$ is the finite path in $\mathcal{N}'$ that results from $\pi_{\mathcal{N}}$ by replacing (from the right to the left) any step $\langle q, w\rangle \xrightarrow{\delta} \langle p, v\rangle$ in $\pi_{\mathcal{N}}$ with $\langle q, w\rangle \xrightarrow{\delta} \langle p_F, v\rangle$ where F is the set of fairness sets f such that $\delta \in f$. Given a scheduler $\mathcal{U}$ for $\mathcal{N}$, then the corresponding scheduler $\mathcal{U}'$ for $\mathcal{N}'$ and the input path $\pi_{\mathcal{N}'}$ behaves as $\mathcal{U}$ for the input path $\pi_{\mathcal{N}}$, i.e., if $\mathcal{U}$ chooses the transition rule $\delta = q \xrightarrow{op} p$ for $\pi_{\mathcal{N}}$ then $\mathcal{U}'$ chooses the transition rule $q_G \xrightarrow{op} p_F$ for $\pi_{\mathcal{N}'}$ where q_G is the location of $\pi_{\mathcal{N}'}$'s last configuration and $F = \{f \in \mathcal{F} : \delta \in f\}$. Then, $\mathcal{U}$ is finite-memory iff $\mathcal{U}'$ is, and $\Pr_{\mathcal{U}'}(s \models \mathit{fair})$ agrees with the probability measure of the $\mathcal{F}$-fair $\mathcal{U}$-paths starting in s. Thus, $\mathcal{U}$ is $\mathcal{F}$-fair iff $\Pr_{\mathcal{U}}(s \models \mathit{fair}) = 1$ for all s.

If $\pi_{\mathcal{N}'}$ is a finite path in $\mathcal{N}'$ then $\pi_{\mathcal{N}'}\uparrow$ denotes the basic cylinder induced by $\pi_{\mathcal{N}'}$, i.e., the set of all infinite paths π' in $\mathcal{N}'$ such that $\pi_{\mathcal{N}'}$ is a prefix of π'. If ψ is an LTL formula and π' and infinite path in $\mathcal{N}'$ then $\pi' \models \psi \wedge \pi_{\mathcal{N}'}$ means $\pi' \models \psi \wedge \pi' \in \pi_{\mathcal{N}'}\uparrow$.

"$\Longleftarrow$": If $\mathcal{U}$ is a $\mathcal{F}$-fair fm-scheduler for $\mathcal{N}$ with $\Pr_{\mathcal{U}}(s \models \varphi) > 0$ then the corresponding scheduler $\mathcal{U}'$ for $\mathcal{N}'$ is finite-memory and fulfills $\Pr_{\mathcal{U}'}(s \models \varphi') > 0$ and $\Pr_{\mathcal{U}'}(s \models \mathit{fair}) = 1$. Hence, $\Pr_{\mathcal{U}'}(s \models \mathit{fair} \wedge \varphi') > 0$.

"$\Longrightarrow$": Let us now assume that $\mathcal{V}$ is a fm-scheduler for $\mathcal{N}'$ such that $\Pr_{\mathcal{V}}(s \models \mathit{fair} \wedge \varphi') > 0$. Let $\mathcal{U}$ be the following fm-scheduler for $\mathcal{N}$. For a given finite path $\pi_{\mathcal{N}}$ in $\mathcal{N}$ with $\Pr_{\mathcal{V}}(s \models \mathit{fair} \wedge \varphi' \wedge \pi_{\mathcal{N}'} \uparrow) > 0$, $\mathcal{U}$ makes the same choice as $\mathcal{V}$ for the corresponding path $\pi_{\mathcal{N}'}$.[3] For all other finite paths $\pi_{\mathcal{N}'}$, $\mathcal{U}$ behaves in arbitrary, but finite-memory $\mathcal{F}$-fair way. We then have $\Pr_{\mathcal{U}}(s \models \varphi) \geq \Pr_{\mathcal{V}}(\mathit{fair} \wedge \varphi') > 0$, since all $\mathcal{V}$-paths π' with $\pi' \models \mathit{fair} \wedge \varphi'$ are also $\mathcal{U}'$-paths for the scheduler $\mathcal{U}'$ associated with $\mathcal{U}$.

It remains to show that $\mathcal{U}$ is $\mathcal{F}$-fair. By definition of $\mathcal{U}$, it suffices to show that almost all infinite $\mathcal{U}$-paths π, where for each finite prefix $\pi_{\mathcal{N}}$ of π, the corresponding path $\pi_{\mathcal{N}'}$ in $\mathcal{N}'$ fulfills $\Pr_{\mathcal{V}}(s \models \mathit{fair} \wedge \varphi' \wedge \pi_{\mathcal{N}'} \uparrow) > 0$, are $\mathcal{F}$-fair. For this, it suffices to show that for each such $\mathcal{U}$-path π, where the corresponding $\mathcal{V}$-path π' in $\mathcal{N}'$ is strongly prob-fair, we have $\pi' \models \mathit{fair} \wedge \varphi'$. Let π be such an infinite $\mathcal{U}$-path and π' the corresponding $\mathcal{V}$-path. We then have $\Pr_{\mathcal{V}}(s \models \mathit{fair} \wedge \varphi' \wedge \pi_{\mathcal{N}'} \uparrow) > 0$ for each finite prefix $\pi_{\mathcal{N}'}$ of π'. Since π' is strongly prob-fair and because $\mathcal{V}$ is finite-memory there exists a finite prefix $\pi_{\mathcal{N}'}$ of π' such that the transition rules taken infinitely often in some path of the basic cylinder $\pi_{\mathcal{N}'} \uparrow$ are also taken infinitely often in π'. As $\Pr_{\mathcal{V}}(s \models \mathit{fair} \wedge \pi_{\mathcal{N}'} \uparrow) > 0$ we get $\pi' \models \mathit{fair}$, and thus, π is $\mathcal{F}$-fair. $\square$

[3] More precisely, the corresonding scheduler $\mathcal{U}'$ behaves as $\mathcal{V}$ for $\pi_{\mathcal{N}'}$. Thus, if $\mathcal{V}$ takes the transition rule $q_G \xrightarrow{op} p_F$ then $\mathcal{U}$ takes the transition rule $q \xrightarrow{op} p$.