Form Methods Syst Des
DOI 10.1007/s10703-012-0168-y

Computable fixpoints in well-structured symbolic model checking

N. Bertrand · P. Schnoebelen

Abstract We prove a general finite-time convergence theorem for fixpoint expressions over a well-quasi-ordered set. This has immediate applications for the verification of well-structured systems, where a main issue is the computability of fixpoint expressions, and in particular for game-theoretical properties and probabilistic systems where nesting and alternation of least and greatest fixpoints are common.

Keywords Verification of well-structured systems · Verification of probabilistic systems · mu-Calculus · Infinite-state systems

1 Introduction

Regular model checking [10, 26, 61, 81] is a popular paradigm for the symbolic verification of models with infinite state space. It has been applied to varied families of systems ranging from parameterized distributed algorithms [4] and channel systems [7] to hybrid systems [23] and list-manipulating programs [24].

In regular model checking, one works with regular sets of states and handles them via finite descriptions, e.g., finite-state automata or regular expressions. Models amenable to regular model checking are such that, when $U \subseteq \mathit{Conf}$ is a "regular" set of configurations, then $\mathit{Post}(U)$ and/or $\mathit{Pre}(U)$, the sets of 1-step successors (respectively, predecessors) of configurations from U, is again a regular set that can be computed effectively from U. Since

This article was written while N. Bertrand was visiting Liverpool Comp. Sci. Dept, and P. Schnoebelen was visiting Oxford Comp. Sci. Dept, supported by the Leverhulme Trust and by Grant ANR-11 BS02-001. An extended abstract of this article appeared in [14].

N. Bertrand
Inria Rennes Bretagne Atlantique, Campus Universitaire de Beaulieu, 35042 Rennes Cedex, France

P. Schnoebelen (✉)
LSV, CNRS & ENS de Cachan, 61, avenue du Président Wilson, 94235 Cachan Cedex, France
e-mail: phs@lsv.ens-cachan.fr

Published online: 04 August 2012

regular sets are closed under Boolean operations, one can try to compute the reachability set $Post^*(Init)$, or the co-reachability set $Pre^*(Final)$, as the limit of the sequences

$$U_0 := Init; \quad U_1 := U_0 \cup Post(U_0); \quad \ldots; \quad U_{n+1} := U_n \cup Post(U_n); \quad \ldots; \tag{1}$$

$$V_0 := Final; \quad V_1 := V_0 \cup Pre(V_0); \quad \ldots; \quad V_{n+1} := V_n \cup Pre(V_n); \quad \ldots. \tag{2}$$

Such computations, essential to any kind of symbolic verification, are possible with any class of representation closed under, and providing algorithms for, *Pre* or *Post*, Boolean operations, vacuity [55, 61].

With infinite-state models, the main difficulty is *convergence*. It is very rare that a fixpoint computation like Eqs. (1)–(2) converges in finite time [18], and innovative techniques that try to compute directly, or guess and check, or approximate the limit sets, are currently under active scrutiny (see, e.g., [18, 20, 22, 25, 53]).

Well-structured transition systems (WSTS) are a generic family of models for which the co-reachability set $Pre^*(Final)$ can be computed symbolically using exactly the sequence (2) [6, 48]. For WSTS's, convergence of the fixpoint computation is ensured by WQO theory: one handles upward-closed sets, and increasing sequences of upward-closed subsets of a WQO (a well-quasi-ordered set) always converge in finite time (see Fact 1 below).

Computing $Pre^*(Final)$ for reachability analysis is just a special case of *fixpoint computation*. When considering richer properties, e.g., temporal logic properties, one is interested in computing more complex fixpoints. Indeed, the set of states where a temporal logic property holds is often definable with *Pre*, Boolean operations, and (sometimes nested) fixpoints [30]. The same holds for game-theoretic and qualitative probabilistic properties, albeit with more complex fixpoint expressions.

Our contribution In this article, we define a notion of μ-expressions where recursion is guarded by upward-closure operators, and give a general finite-time convergence theorem for all such expressions when evaluated over the powerset of a WQO. The consequence is that these fixpoint expressions can be evaluated symbolically by an iterative procedure. The guarded fragment we isolate is very relevant for the verification of well-structured transition systems. We illustrate this point by providing direct proofs of decidability results on several classes of WSTS models, ranging from monotonic counter systems to probabilistic lossy channel systems.

Related work Henzinger *et al.* give general conditions for the convergence of fixpoints computations for temporal [55] or game-theoretic [35] properties, but the underlying framework is different: it relies on finite quotients and mainly aims at timed and hybrid systems. For WSTS's, a generic computability result for a fragment of the μ-language we consider is briefly mentioned in [63]. Our applications to well-structured transition systems generalize results from [2, 5, 65, 72, 73] that rely on more ad-hoc proofs of convergence.

Outline The first part of this article, Sects. 2–4, presents our framework and our main theorems in a generic way, using simple explanatory examples to illustrate the main ideas. Applications to the verification of well-structured transition systems are covered in a second part, Sects. 5–9.

2 A mu-calculus for symbolic verification

Symbolic verification of infinite-state systems can be formalized in several ways (among other possibilities let us mention automatic structures [62], decidable logics, and the abstract interpretation framework [34]).

In this article, we adopt a simple generic approach, called "monotonic region algebra", that is sufficient for our purposes. It is based on a simple algebra of computable sets that is embedded in the powerset (a complete lattice) of a WQO, and where only monotonic operations on sets are considered.

We start by recalling some classic notions from WQO theory [64, 69]: A quasi-ordering over a set W is a reflexive and transitive relation $\sqsubseteq \subseteq W \times W$. A quasi-ordering is a *well-quasi-ordering* (a WQO) if it admits no *infinite antichains*, i.e., infinite subsets of mutually incomparable elements, and is *well-founded*, i.e., admits no infinite strictly decreasing sequences. Equivalently, $\sqsubseteq$ is a WQO iff any infinite sequence $w_1, w_2, w_3, \ldots$ of elements of W contains an infinite increasing subsequence of the form $w_{i_1} \sqsubseteq w_{i_2} \sqsubseteq w_{i_3} \sqsubseteq \cdots$ (where $i_1 < i_2 < i_3 < \cdots$). In the rest of this article, we assume that $(W, \sqsubseteq)$ is a well-quasi-ordered set, i.e., a set equipped with a WQO.

We say that a subset V of W is *upward-closed* if $v \in V$ and $v \sqsubseteq w$ implies $w \in V$. Our developments rely on the Finite-Time Convergence Property, a basic fact from WQO theory, stating that any infinite increasing sequence of upward-closed subsets of a WQO stabilizes after finitely many steps.

Fact 1 (Finite-Time Convergence Property) *Let* $V_0 \subseteq V_1 \subseteq V_2 \subseteq \cdots$ *be an infinite* increasing *sequence of* upward-closed *subsets of a WQO* W. *Then for some index* $k \in \mathbb{N}$, $V_k = V_{k+1} = V_{k+2} = \cdots = \bigcup_{i \in \mathbb{N}} V_i$.

In algebraic terminology, this simply states that upward-closed subsets of W satisfy the Ascending Chain Condition, and is another *characterization* of WQO's (see, e.g., [60, Theorem 1.2]).

There is a symmetric notion of *downward-closed* subsets of W. Since the complement of an upward-closed subset is downward-closed and *vice versa*, the Finite-Time Convergence Property also states that, dually, an infinite *decreasing* sequence of *downward-closed* subsets of W eventually stabilizes.

Remark 2 (On complexity and the time to convergence) The complexity analysis of an algorithm whose termination relies on the Finite-Time Convergence Property requires some bound on the index k at which stabilization is achieved. In this article, we focus on generic decidability issues. The concluding section discusses complexity and feasibility issues and provides relevant pointers.

Given a set $V \subseteq W$, we write $C_\uparrow(V)$ for its *upward-closure*, i.e., the smallest upward-closed set containing V, and $K_\uparrow(V)$ for its *upward-interior*, i.e., the largest upward-closed set included in V. The *downward-closure* of V, written $C_\downarrow(V)$ and its *downward-interior*, written $K_\downarrow(V)$, are defined analogously. Observe that $C_\uparrow$ and $C_\downarrow$ are extensive, $K_\uparrow$ and $K_\downarrow$ are contractive, and the following dualities hold:

$$W \setminus K_\uparrow(V) = C_\downarrow(W \setminus V), \qquad W \setminus K_\downarrow(V) = C_\uparrow(W \setminus V). \tag{3}$$

2.1 Monotonic region algebra

Let $O = \{o_1, o_2, \ldots\}$ be a countable set of *operator names* equipped with an *arity* function $ar : O \to \mathbb{N}$. Here $ar(o)$ is the number of arguments taken by o, and the pair (O, ar) is called a *signature*. When $ar(o) = 0$, o is called a *constant*.

Definition 3 A *monotonic region algebra* over W is a structure $\mathcal{R} = \langle W, \mathbf{R}; (o^{\mathcal{R}})_{o \in O} \rangle$ where

1. $\mathbf{R} \subseteq 2^W$ is a set of distinguished subsets of W, called *regions*, that contains in particular $\emptyset$ and W, and
2. for every $k \in \mathbb{N}$ and every $o \in O$ with $ar(o) = k$, the interpretation $o^{\mathcal{R}}$ of o is a k-ary map $o^{\mathcal{R}} : (2^W)^k \to (2^W)$ on subsets of W that satisfies:
   - **monotonicity** w.r.t. set-inclusion, i.e.,

$$U_1 \subseteq V_1 \wedge \cdots \wedge U_k \subseteq V_k \quad \text{implies} \quad o^{\mathcal{R}}(U_1, \ldots, U_k) \subseteq o^{\mathcal{R}}(V_1, \ldots, V_k), \tag{4}$$

   - **preservation** of regions, i.e.,

$$U_1, \ldots, U_k \in \mathbf{R} \quad \text{implies} \quad o^{\mathcal{R}}(U_1, \ldots, U_k) \in \mathbf{R} \tag{5}$$

   for all $U_1, \ldots, U_k, V_1, \ldots, V_k \in 2^W$.

As a special case of Eq. (5), constants must be interpreted as regions: $o^{\mathcal{R}} \in \mathbf{R}$ when $ar(o) = 0$. For simplicity, we often just write o instead of $o^{\mathcal{R}}$ when this causes no confusion.

When $ar(o) = n = ar(o')$, we write $o^{\mathcal{R}} \leq o'^{\mathcal{R}}$, or just $o \leq o'$, when $o^{\mathcal{R}}(U_1, \ldots, U_n) \subseteq o'^{\mathcal{R}}(U_1, \ldots, U_n)$ for all $U_1, \ldots, U_n \subseteq W$. A unary operator $o^{\mathcal{R}}$ is *extensive* if $U \subseteq o^{\mathcal{R}}(U)$ for all U. It is *contractive* if $o^{\mathcal{R}}(U) \subseteq U$.

We say that a monotonic region algebra is *effective* when, informally, the $o^{\mathcal{R}}$ operations restricted on regions are recursive, and some fundamental predicates like vacuity and equality of regions, are decidable. We do not want to make the definition more pedantically formal but of course this assumes that $\mathbf{R}$ is countable, that an index system (or a data structure) exists for denoting the sets in $\mathbf{R}$, and that the recursive functions implementing the operations in O are given uniformly when O is infinite. In practice, one only considers a finite set of operations (that may admit extra parameters) and typically uses data structures inspired by automata theory or constraint solving.

Example 4 (Subword ordering and regular regions) Let $\Sigma = \{a, b, \ldots\}$ be a finite alphabet. We say that a word $x \in \Sigma^*$ is a (scattered) subword of $y \in \Sigma^*$, written $x \sqsubseteq y$, $\overset{\text{def}}{\Leftrightarrow}$ there exists a factorization $x = x_1 \ldots x_n$ of x and padding words $z_0, \ldots, z_n \in \Sigma^*$ such that $y = z_0 x_1 z_1 x_2 \ldots z_{n-1} x_n z_n$. In other words, x can be obtained by erasing some letters from y. It is well known that $(\Sigma^*, \sqsubseteq)$ is a WQO when Σ is finite (Higman's Lemma). A standard example of a monotonic region algebra over $W = \Sigma^*$ is obtained by choosing all regular languages as regions: we denote this algebra with $\mathcal{R}_{\mathbf{Reg}}(\Sigma) \overset{\text{def}}{=} \langle \Sigma^*, \mathbf{Reg}(\Sigma); \ldots \rangle$ where the set of monotonic operators is left implicit (and will vary with applications). It is an easy exercise to show that the closure operators preserve regions (see [52] for efficient algorithms when R is given by a NFA), and that the interior operators are region-preserving too since the complement of $R \subseteq \Sigma^*$ is regular when R is.

The signature can include any monotonic region-preserving operators. Most standard operations on languages, e.g., concatenation $L_1.L_2$, shuffle $L_1 \parallel L_2$, mirroring $\overleftarrow{L}$, star-closure L^*, left- and right-residuals ($L^{-1}L' \stackrel{\text{def}}{=} \{v \mid \exists u \in L, uv \in L'\}$), conjugacy ($\widetilde{L} \stackrel{\text{def}}{=} \{vu \mid uv \in R\}$), homomorphic and inverse-homomorphic images, are *collecting*, i.e., lifted from functions or relations on words, according to the pattern $o(L_1, L_2) \stackrel{\text{def}}{=} \{o'(u, v) \mid u \in L_1, v \in L_2\}$ or some variant. Such languages operators are always monotonic, and the main issue is whether they preserve regularity. This is the case for all the examples above, and many more [70]. Complementation is not allowed because it is not monotonic, but this is not a true limitation in practice since the dual $\widetilde{o}(L_1, \ldots) \stackrel{\text{def}}{=} \Sigma^* \setminus o(\Sigma^* \setminus L_1, \ldots)$ of any monotonic regularity-preserving operator is itself monotonic and regularity-preserving, hence can be added to the signature.

Finally, regarding effectiveness, it is well known that all the operators we mentioned are effective for regular regions represented in any of the standard ways, e.g., via FSA's or regular expressions.

Example 5 (Subword ordering and computability of Higman-Haines languages) A general fact about WQO's is that the upward and downward closures of arbitrary sets have simple finite representations [46]. In the case of $(\Sigma^*, \sqsubseteq)$, the closures of *arbitrary languages*, called Higman-Haines sets, and their interiors, are always regular. However this is usually not effective [42, 52].

Consider for example $\mathcal{R}_{\mathbf{CF}}(\Sigma)$, the region algebra having context-free languages $R \in \mathbf{CF}(\Sigma)$ as regions. It accommodates closure and interior operators since they return regular languages and $\mathbf{CF}(\Sigma)$ contains $\mathbf{Reg}(\Sigma)$. It accommodates other region-preserving operators, e.g., set-union, concatenation and star-closure in its signature, but not set-intersection since $R, R' \in \mathbf{CF}(\Sigma)$ does not imply $R \cap R' \in \mathbf{CF}(\Sigma)$.

Regarding effectiveness, (FSA's for) the closures $C_\uparrow(R)$ and $C_\downarrow(R)$ can be computed effectively from (a context-free grammar for) $R \in \mathbf{CF}(\Sigma)$ [80]. However, the interior operators are not effective. Indeed, universality is undecidable for context-free grammars but decidable for regular languages, and $R = \Sigma^*$ iff $K_\uparrow(R) = K_\downarrow(R) = \Sigma^*$.

Example 6 (Natural numbers and semilinear sets) The set $\mathbb{N}^d$ of all d-dimensional vectors of natural numbers is naturally ordered by the *product ordering*: for any $\mathbf{a} = (a_1, \ldots, a_d)$ and $\mathbf{b} = (b_1, \ldots, b_d)$, $\mathbf{b} \leq \mathbf{a} \stackrel{\text{def}}{\Leftrightarrow} a_1 \leq b_1 \wedge \cdots \wedge a_d \leq b_d$. It is well known that $(\mathbb{N}^d, \leq)$ is a WQO when d is finite (Dickson's Lemma). A standard monotonic region algebra over $W = \mathbb{N}^d$, denoted $\mathcal{R}_{\mathbf{SL}}(d)$ in the sequel, is obtained by choosing as regions all semilinear subsets, or equivalently, all subsets definable in Presburger arithmetic. Many natural operators on sets of vectors, including the closure and interior operators, can be defined in Presburger arithmetic, hence they are region-preserving. In practice, many of them are collecting and hence monotonic.

In the rest of this article, we always assume that O contains the unary operators $C_\uparrow, C_\downarrow, K_\uparrow, K_\downarrow$, and that these requisite operators are given their standard WQO-theoretical interpretation.

2.2 Region algebra with fixpoints

We now extend the symbolic framework with least and greatest fixpoints. Let $\chi = \{X, Y, \ldots\}$ be a countable set of variables. We write $L_\mu(O)$, or shortly L_μ when O is understood, for

the set of *O-terms with least and greatest fixpoints*, given by the following abstract syntax:

$$\varphi ::= o(\varphi_1, \ldots, \varphi_k) \mid X \mid \mu X.\varphi \mid \nu X.\varphi$$

where X runs over variables from χ, and o over k-ary operators from O. Terms $\mu X.\varphi$ and $\nu X.\varphi$ are fixpoint expressions, classically used in μ-calculi [12]. We make the standard assumption that no variable has both bound and free occurrences in some φ, and that no two fixpoint subterms bind the same variable: this can always be ensured by renaming bound variables.

As is standard, we write $\varphi(X_1, \ldots, X_n)$ to stress that the free variables occurring in φ are among $X_1, \ldots, X_n$, and we use $\varphi(\psi_1, \ldots, \psi_n)$ to denote the term obtained from $\varphi(X_1, \ldots, X_n)$ by replacing all occurrences of the X_i's by the corresponding ψ_i's terms.

Assuming a monotonic region algebra $\mathcal{R}$, the interpretation of L_μ terms is as expected: a term $[\![\varphi(X_1, \ldots, X_n)]\!]$ denotes a mapping from $(2^W)^{\{X_1,\ldots,X_n\}}$ to 2^W. In other words, given an *environment* $env : \{X_1, \ldots, X_n\} \to 2^W$ that associates a subset of W with each X_i free in φ, $[\![\varphi]\!](env)$, more simply denoted $[\![\varphi]\!]_{env}$, is a subset of W. We often consider that $[\![\varphi]\!]$ has type $(2^W)^n \to 2^W$ and write $[\![\varphi]\!](U_1, \ldots, U_n)$ —or $[\![\varphi]\!]$ when $n = 0$— instead of $[\![\varphi]\!]_{env}$ (where $env(X_i) = U_i$): this only assumes that the correspondence between free variables and arguments of $[\![\varphi]\!]$ is understood.

Definition 7 (Semantics of L_μ) $[\![\varphi]\!]_{env}$ is defined by structural induction:

$$[\![X]\!]_{env} \stackrel{\text{def}}{=} env(X), \qquad [\![o(\varphi_1, \ldots, \varphi_k)]\!]_{env} \stackrel{\text{def}}{=} o^{\mathcal{R}}\big([\![\varphi_1]\!]_{env}, \ldots, [\![\varphi_k]\!]_{env}\big),$$

$$[\![\mu X.\varphi]\!]_{env} \stackrel{\text{def}}{=} \mathbf{lfp}\big(\Omega[\varphi, X, env]\big), \qquad [\![\nu X.\varphi]\!]_{env} \stackrel{\text{def}}{=} \mathbf{gfp}\big(\Omega[\varphi, X, env]\big),$$

$$\text{where } \Omega[\varphi, X, env](U) \stackrel{\text{def}}{=} [\![\varphi]\!]_{env \oplus [X \mapsto U]}.$$

Here $env \oplus [X \mapsto U]$ denotes the environment obtained by extending env to one more variable, while $\mathbf{lfp}(\Omega[\ldots])$ and $\mathbf{gfp}(\Omega[\ldots])$ are the least and greatest fixpoints of a unary mapping $\Omega[\varphi, X, env] : 2^W \to 2^W$ that could be informally defined with $\Omega(V) = [\![\varphi]\!](V, env(X_1), \ldots, env(X_n))$.

Observe that the semantics of the fixpoint terms is well defined. Indeed, $(2^W, \subseteq)$ is a complete lattice and every $[\![\varphi(X_1, \ldots, X_n)]\!]$ is monotonic in its n arguments (this is shown by induction on the structure of φ) so that $\Omega[\varphi, X, env]$, being $[\![\varphi]\!]$ with some arguments already fixed, is monotonic in its remaining argument and has well-defined least and greatest fixpoints (Knaster-Tarski Theorem).

Moreover, $U_1 \subseteq V_1, \ldots, U_n \subseteq V_n$ implies $[\![\varphi]\!](U, U_1, \ldots, U_n) \subseteq [\![\varphi]\!](U, V_1, \ldots, V_n)$ by monotonicity of $[\![\varphi]\!]$. Hence, if $env(X_i) = U_i$ and $env'(X_i) = V_i$ for $i = 1, \ldots, n$, then $\Omega[\varphi, X, env](U) \subseteq \Omega[\varphi, X, env'](U)$, written more simply $\Omega \preceq \Omega'$. This entails $\mathbf{lfp}(\Omega) \subseteq \mathbf{lfp}(\Omega')$ (i.e., $[\![\mu X.\varphi]\!](U_1, \ldots, U_n) \subseteq [\![\mu X.\varphi]\!](V_1, \ldots, V_n)$) and $\mathbf{gfp}(\Omega) \subseteq \mathbf{gfp}(\Omega')$. In other words, $[\![\mu X.\varphi]\!]$ and $[\![\nu X.\varphi]\!]$ are monotonic.

We now make a crucial observation: for an L_μ term $\varphi(X_1, \ldots, X_n)$ and regions $U_1, \ldots, U_n$, the set $[\![\varphi]\!](U_1, \ldots, U_n)$ is in general not a region. That is, $[\![\varphi]\!]$ is usually not region-preserving (and not computable even when $\mathcal{R} = \langle W, \mathbf{R}; \ldots \rangle$ is effective). This is because $\mathbf{R}$ is usually not a complete sub-lattice of 2^W. At the moment, only the terms φ that do not use fixpoints can be guaranteed to have $[\![\varphi]\!] \in \mathbf{R}$. In the next section, we present a larger fragment of L_μ for which $[\![\varphi]\!]$ is guaranteed to be region-preserving (and computable).

Example 8 (Fixpoints in the algebra of regular languages) Consider the region algebra $\mathcal{R}_{\mathbf{Reg}}(\Sigma)$ from Example 4. Using concatenation, set-union and constants, one can write terms like $\mu X.\varepsilon + a \cdot X \cdot b$ whose value is $\{a^n b^n \mid n \in \mathbb{N}\} \notin \mathbf{Reg}(\Sigma)$. With least and greatest fixpoints, and when a few more primitive (monotonic) operators like intersection, morphisms, etc., are allowed, one can describe complex languages, well beyond the recursive ones [59, 66].

We abuse notation and write $\varphi(X_1, \ldots, X_n) \leq \varphi'(X_1, \ldots, X_n)$ when $[\![\varphi]\!] \leq [\![\varphi']\!]$. Note that $\varphi \leq \varphi'$ entails $\mu X.\varphi \leq \mu X.\varphi'$. We say that φ and φ' are *equivalent*, and write $\varphi = \varphi'$, when they denote the same mapping, or equivalently when $\varphi \leq \varphi' \leq \varphi$.

In the following, and in particular in the application sections, we often transform L_μ terms into equivalent ones that are more convenient. This is done using simple logical or algebraic transformations, e.g., lattice-theoretical properties like distributivity of $\cap$ w.r.t. $\cup$, etc. (This assumes that $\cap$ and $\cup$ appear in the signature O and have their set-theoretical meaning.) Some of the transformations we use are more specific to μ-calculi, and we list them here for the sake of completeness.

Lemma 9 (Some useful μ-calculus laws)

- ***Unfolding***: *for any* $\varphi(X, Y_1, \ldots, Y_n)$ *in* L_μ,

$$\begin{aligned} \mu X.\varphi(X, \ldots) &= \varphi\big(\mu X.\varphi(X, \ldots), \ldots\big) = \mu X.\varphi\big(\varphi(X, \ldots), \ldots\big), \\ \nu X.\varphi(X, \ldots) &= \varphi\big(\nu X.\varphi(X, \ldots), \ldots\big) = \nu X.\varphi\big(\varphi(X, \ldots), \ldots\big). \end{aligned} \tag{6}$$

- ***Fixpoint rule***: *for any* $\varphi(X, Y_1, \ldots, Y_n)$ *and* $\psi(Y_1, \ldots, Y_n)$ *in* L_μ,

$$\varphi\big(\psi(Y_1, \ldots), Y_1, \ldots\big) \leq \psi(Y_1, \ldots) \quad \textit{implies} \quad \mu X.\varphi(X, Y_1, \ldots) \leq \psi(Y_1, \ldots). \tag{7}$$

- ***Extensive/contractive fixpoints***: *for any* $\varphi(X, \ldots)$ *and* $\psi(X, \ldots)$ *in* L_μ,

$$\mu X.(X \cap \psi) \cup \varphi = \mu X.\varphi, \qquad \nu X.(X \cup \psi) \cap \varphi = \nu X.\varphi, \tag{8}$$

  while for any $\varphi(X, Y, \ldots)$ *in* L_μ,

$$\mu Y.\nu X.(Y \cup \varphi) = \mu Y.\nu X.\varphi, \qquad \nu Y.\mu X.(Y \cap \varphi) = \nu Y.\mu X.\varphi. \tag{9}$$

- ***Commutation of extensive fixpoints***: *if* f, g *are* extensive *unary operators*,

$$\mu X.f\big(g(X)\big) = \mu X.\big[f(X) \cup g(X)\big] = \mu X.g\big(f(X)\big). \tag{10}$$

Proof [Sketch] The unfolding equalities and the fixpoint rule are well known [12]. We prove equalities (8) and (10) for the sake of illustration. Equation (9)—see [19] for a proof—is an extension of (8) that we use in Sect. 9.

For Eq. (8), let $\varphi'(X, \ldots) \stackrel{\text{def}}{=} (X \cap \psi) \cup \varphi(X, \ldots)$. Now $\varphi'(\mu X.\varphi, \ldots) = (\mu X.\varphi) \cap \psi \cup \varphi(\mu X.\varphi, \ldots)$ by def. of φ', $= (\mu X.\varphi) \cap \psi \cup \mu X.\varphi$ by Eq. (6), $= \mu X.\varphi$. The fixpoint rule applies and yields $\mu X.\varphi' \leq \mu X.\varphi$. But $\varphi \leq \varphi'$ by definition, so that $\mu X.\varphi \leq \mu X.\varphi'$. Finally, $\mu X.\varphi = \mu X.\varphi' = \mu X.X \cap \psi \cup \varphi$. The other half of Eq. (8) is entailed by duality.

For Eq. (10), let $h(X) \stackrel{\text{def}}{=} f(X) \cup g(X)$. Since f, g and h are extensive, and since $f \leq h$ and $g \leq h$, we deduce that $h \leq f \circ g \leq h \circ h$, hence $\mu X.h(X) \leq \mu X.f(g(X)) \leq \mu X.h(h(X))$ by monotonicity. On the other hand $\mu X.h(h(X)) = \mu X.h(X)$ by Eq. (6). Thus $\mu X.h(x) = \mu X.f(g(X))$. □

3 Guarded terms and finite-time convergence

We now come to the concept of guardedness by closure and interior operators.

Definition 10 (Guarded variables and guarded terms) Let $\varphi \in L_\mu$.

1. A variable X is *upward-guarded* in φ if each of its free occurrences in φ is under the scope[1] of one of the upward operators $C_\uparrow$ or $K_\uparrow$, i.e., appears in a subterm of the form $C_\uparrow(\psi)$ or $K_\uparrow(\psi)$.
2. A variable X is *downward-guarded* in φ if, symmetrically, each of its free occurrences in φ is under the scope of $C_\downarrow$ or $K_\downarrow$.
3. The term φ is *guarded* if all its subterms of the form $\mu X.\psi$ have X upward-guarded in ψ, and all its subterms of the form $\nu X.\psi$ have X downward-guarded in ψ.

Guardedness ensures that fixpoint terms can be computed via their approximants. Formally, for φ, X and *env*, the *approximants* of $[\![\mu X.\varphi]\!]_{env}$ are the sequence $(M_i)_{i\in\mathbb{N}}$ of subsets of W defined inductively with $M_0 = \emptyset$ and $M_{i+1} = [\![\varphi]\!]_{env\oplus[X\mapsto M_i]}$. Similarly, the *approximants of* $[\![\nu X.\varphi]\!]_{env}$ are the sequence $(N_i)_{i\in\mathbb{N}}$ defined by $N_0 = W$ and $N_{i+1} = [\![\varphi]\!]_{env\oplus[X\mapsto N_i]}$. Writing simply Ω for $\Omega[\varphi, X, env]$, M_i is $\Omega^i(\emptyset)$ and N_i is $\Omega^i(W)$. These approximants are linearly ordered:

$$\begin{aligned} &M_0 \subseteq M_1 \subseteq M_2 \subseteq \cdots \subseteq \mathbf{lfp}(\Omega) = [\![\mu X.\varphi]\!]_{env}, \\ &N_0 \supseteq N_1 \supseteq N_2 \supseteq \cdots \supseteq \mathbf{gfp}(\Omega) = [\![\nu X.\varphi]\!]_{env}. \end{aligned} \tag{11}$$

Lemma 11 (Finite-time convergence of approximants) *For $\varphi \in L_\mu$, let $(M_i)_{i\in\mathbb{N}}$ and $(N_i)_{i\in\mathbb{N}}$ be, respectively, the approximants of $[\![\mu X.\varphi]\!]_{env}$ and $[\![\nu X.\varphi]\!]_{env}$.*

If X is upward-guarded in φ, then there exists an index $k \in \mathbb{N}$ such that

$$[\![\mu X.\varphi]\!]_{env} = M_k = M_{k+1} = M_{k+2} = \cdots. \tag{12}$$

Dually, if X is downward-guarded in φ, then there exists a $k \in \mathbb{N}$ such that

$$[\![\nu X.\varphi]\!]_{env} = N_k = N_{k+1} = N_{k+2} = \cdots. \tag{13}$$

Proof We only prove the first half since the other half is dual. Let $\psi_1, \ldots, \psi_m$ be the maximal subterms of φ that are immediately under the scope of a $C_\uparrow$ or a $K_\uparrow$ operator. Then φ can be decomposed under the form

$$\varphi \equiv \Phi(\Uparrow \psi_1, \ldots, \Uparrow \psi_m)$$

where $\Phi(Y_1, \ldots, Y_m)$ is a context with fresh variables $Y_1, \ldots, Y_m$, and where, for $j = 1, \ldots, m$, $\Uparrow \psi_j$ is either $C_\uparrow(\psi_j)$ or $K_\uparrow(\psi_j)$, depending on how ψ_j appears in φ. In either case, and for any environment env', the set $[\![\Uparrow \psi_j]\!]_{env'}$ is upward-closed.

Since X is upward-guarded in φ, it has no occurrence in $\Phi(Y_1, \ldots, Y_m)$, only in the ψ_j's, so that

$$\begin{aligned} M_{i+1} = [\![\varphi]\!]_{env\oplus[X\mapsto M_i]} &= [\![\Phi]\!]\big([\![\Uparrow \psi_1]\!]_{env\oplus[X\mapsto M_i]}, \ldots, [\![\Uparrow \psi_m]\!]_{env\oplus[X\mapsto M_i]}\big) \\ &= [\![\Phi]\!](L_{i,1}, \ldots, L_{i,m}) \end{aligned}$$

[1] Note that the occurrence of X is not required to be under the *immediate* scope of a closure or interior operator. See the guarded term in Example 13.

writing $L_{i,j}$ for $[\![\Uparrow \psi_j]\!]_{env \oplus [X \mapsto M_i]}$. Note that $L_{i,j}$ is upward-closed. By monotonicity of $[\![\Uparrow \psi_j]\!]$, $M_0 \subseteq M_1 \subseteq M_2 \subseteq \cdots$ entails $L_{0,j} \subseteq L_{1,j} \subseteq L_{2,j} \subseteq \cdots$. Since the $L_{i,j}$'s are upward-closed, for each $j = 1, \ldots, m$, there is an index k_j such that $L_{i,j} = L_{k_j,j}$ for all $i \geq k_j$ (Fact 1). Picking $K = \max(k_1, \ldots, k_m)$ gives for any $i \geq K$

$$\begin{aligned} M_{i+1} = [\![\Phi]\!](L_{i,1}, \ldots, L_{i,m}) &= [\![\Phi]\!](L_{k_1,1}, \ldots, L_{k_m,m}) \\ &= [\![\Phi]\!](L_{K,1}, \ldots, L_{K,m}) = M_{K+1}. \end{aligned}$$

Thus, $\bigcup_{i \in \mathbb{N}} M_i = M_{K+1} = M_{K+2}$ and M_{K+1} is a fixpoint of $\Omega[\varphi, X, env]$, hence the least one thanks to Eq. (11). Picking $k = K + 1$ satisfies Eq. (12). □

Lemma 11 is the key to our main theorem:

Theorem 12 (Guarded terms are computable) *If $\varphi(X_1, \ldots, X_n) \in L_\mu$ is guarded then $[\![\varphi]\!]$ is a region-preserving (monotonic) function. Furthermore, if the region algebra is effective, then $[\![\varphi]\!]$ is a computable function (over regions).*

Proof The proof is by structural induction on φ. For this, observe that if φ is guarded, all its subterms are guarded too. We consider four cases.

1. If $\varphi = X$ is a variable, $[\![\varphi]\!]$ is the identity function, Id_W, and is region-preserving.
2. If φ is some $o(\varphi_1, \ldots, \varphi_k)$, the $[\![\varphi_i]\!]$'s are (computable) region-preserving by induction hypothesis. By definition, $o^{\mathcal{R}}$ too is region-preserving (and computable when the region algebra is effective). Then $[\![\varphi]\!]$, being the composition of $o^{\mathcal{R}}$ and the $[\![\varphi_i]\!]$'s, is region-preserving (and computable). Note that this includes the case where o is nullary (is a constant): $[\![o]\!]$ is a (computable) region.
3. If φ is some $\mu X.\psi(X, X_1, \ldots, X_n)$, we consider any environment env such that $env(X_j)$ is a region for $j = 1, \ldots, n$ and prove by induction on i that each approximant M_i of $[\![\varphi]\!]_{env}$ is a region. $M_0 = \emptyset$ is a region by definition, and if M_i is a region, then $M_{i+1} = [\![\psi]\!](M_i, env(X_1), \ldots, env(X_n))$ is one too, since by induction hypothesis $[\![\psi]\!]$ is region-preserving. We conclude that $[\![\varphi]\!]_{env}$, being some M_k's by Lemma 11, is a region. When $\mathcal{R}$ is effective, the M_i's can be computed effectively, and one can detect when $M_k = M_{k+1}$ since region equality is decidable by assumption. Then $[\![\varphi]\!]_{env} = M_k$ can be computed effectively.
4. If φ is some $\nu X.\psi$, the reasoning is similar to the previous case.

□

Example 13 (Guarded fixpoints in the algebra of regular languages) We continue Examples 4 and 8. While L_μ-terms can describe very complex languages, guarded terms are guaranteed to describe regular languages. E.g., with $\Sigma = \{a, b\}$, the language defined by $\nu X.K_\downarrow(\varepsilon + a + b + a \cdot X \cdot a + b \cdot X \cdot b)$ (inspired by the definition of palindromes) must be regular since it is some $[\![\nu X.\psi(X)]\!]$ where X is downward-guarded in ψ.

Furthermore, this language can be computed effectively as a limit of approximants. One starts with $N_0 = \Sigma^*$. To compute $N_1 = [\![\psi]\!](N_0) = K_\downarrow(\varepsilon + a + b + a \cdot N_0 \cdot a + b \cdot N_0 \cdot b)$, we use Eq. (3), or "$K_\downarrow = \neg C_\uparrow \neg$": the complement of $\varepsilon + a + b + a\Sigma^* a + b\Sigma^* b$ is $a\Sigma^* b + b\Sigma^* a$, whose upward-closure is $C_\uparrow(ab) + C_\uparrow(ba)$, whose complement is $N_1 = a^* + b^*$. Then $N_2 \stackrel{\text{def}}{=} [\![\psi]\!](N_1) = a^* + b^* = N_1$: the fixpoint $[\![\nu X.\psi(X)]\!]$ was reached after finitely many steps.

4 Systems of fixpoint equations

Fixpoint *equations* are an alternative way of writing μ-calculus terms. This presentation can make nested fixpoints easier to read, and it can simplify a large term by eliminating the need to duplicate repeated identical subterms. In this section we explain how to adapt the notion of guardedness for this setting in order to extend Theorem 12.

We start by recalling the necessary notations and definitions (see also [12]).

Definition 14 An n-dimensional *system of fixpoint equations* is a sequence of the form

$$\begin{aligned} X_1 &\overset{\lambda_1}{=} \varphi_1(X_1, \ldots, X_n, Y_1, \ldots, Y_m), \\ &\vdots \\ X_n &\overset{\lambda_n}{=} \varphi_n(X_1, \ldots, X_n, Y_1, \ldots, Y_m), \end{aligned}$$

where, for $i = 1, \ldots, n$, λ_i is either μ or ν, and where $\varphi_1, \ldots, \varphi_n$ are L_μ terms.

Such a system is often denoted in the shorter vector form $\vec{X} \overset{\Lambda}{=} \vec{\varphi}(\vec{X}, \vec{Y})$, where Λ is a vector of n symbols μ or ν. It defines a unique mapping from $(2^W)^{\vec{Y}}$ to $(2^W)^n$—or more simply from $(2^W)^m$ to $(2^W)^n$—, called its *solution*, and denoted $[\![\Lambda\vec{X}.\vec{\varphi}]\!]$.

For $env \in (2^W)^{\vec{Y}}$, the definition of $[\![\Lambda\vec{X}.\vec{\varphi}]\!]_{env}$ extends Definition 7 and is by induction on n. When $n = 0$, $[\![\Lambda\vec{X}.\vec{\varphi}]\!]_{env}$ is the empty tuple $\langle\rangle$. For $n > 0$, $[\![\Lambda\vec{X}.\vec{\varphi}]\!]_{env}$ is the tuple $\langle U_1, \ldots, U_n\rangle$ given by

$$\langle U_2, \ldots, U_n\rangle \overset{\text{def}}{=} \left[\!\!\left[\begin{array}{l} X_2 \overset{\lambda_2}{=} \varphi_2(\psi_1(X_2, \ldots, X_n, \vec{Y}), X_2, \ldots, X_n, \vec{Y}) \\ \quad\vdots \\ X_n \overset{\lambda_n}{=} \varphi_n(\psi_1(X_2, \ldots, X_n, \vec{Y}), X_2, \ldots, X_n, \vec{Y}) \end{array} \right]\!\!\right]_{env}, \tag{14}$$

$$U_1 \overset{\text{def}}{=} [\![\psi_1(X_2, \ldots, X_n, \vec{Y})]\!]_{env \oplus [X_2 \mapsto U_2, \ldots, X_n \mapsto U_n]}, \tag{15}$$

where

$$\psi_1(X_2, \ldots, X_n, \vec{Y}) \overset{\text{def}}{=} \lambda_1 X_1.\varphi_1(X_1, \ldots, X_n, \vec{Y}). \tag{16}$$

Remark 15 (On the semantics of mutually recursive equations) In the special case where $n = 1$, the above definition gives $[\![\psi_1(\vec{Y})]\!]$, i.e., $[\![\lambda_1 X_1.\varphi_1(X_1, \vec{Y})]\!]$ as a solution, in accordance with the semantics of L_μ terms. More generally, it can be seen as a kind of Gaussian elimination that transforms any system of n mutually recursive fixpoint equations into an equivalent non-recursive definition "$X_1 = \Phi_1(\vec{Y}), \ldots, X_n = \Phi_n(\vec{Y})$" based on usual L_μ terms.

Example 16 Consider the following system

$$X_1 \overset{\mu}{=} Y_1 \cup X_1 \cup X_2, \qquad X_2 \overset{\nu}{=} Y_2 \cap X_1 \cap X_2, \tag{17}$$

where we assume that $\cup$ and $\cap$ have their standard set-theoretical meaning.

One solves this system by first letting $\psi_1(X_2, Y_1, Y_2) \stackrel{\text{def}}{=} \mu X_1.(Y_1 \cup X_1 \cup X_2)$. This simplifies as $\psi_1 = Y_1 \cup X_2$ with Eq. (8). Now one rewrites the second equation as

$$X_2 \stackrel{\nu}{=} Y_2 \cap \psi_1(X_2, Y_1, Y_2) \cap X_2, \quad \text{that is,} \quad X_2 \stackrel{\nu}{=} Y_2 \cap (Y_1 \cup X_2) \cap X_2,$$

whose solution is $[\![\nu X_2.Y_2 \cap (Y_1 \cup X_2) \cap X_2]\!]$, or more simply $[\![\nu X_2.Y_2]\!] = [\![Y_2]\!]$ using Eq. (8). Now $\psi_1(X_2, Y_1, Y_2)$ simplifies as $Y_1 \cup Y_2$ and the whole system is equivalent to "$X_1 = Y_1 \cup Y_2$, $X_2 = Y_2$".

Observe that the order in which the fixpoint equations are listed in a system is relevant when it comes to defining what takes precedence among the mixed least and greatest fixpoint selectors. In effect, the higher-numbered λ_i's have priority over the lower-numbered ones. If we now reverse the order of the equations, we get a system

$$X_2 \stackrel{\nu}{=} Y_2 \cap X_1 \cap X_2, \qquad X_1 \stackrel{\mu}{=} Y_1 \cup X_1 \cup X_2, \tag{17′}$$

solved by writing $\psi_2'(X_1, Y_1, Y_2) \stackrel{\text{def}}{=} \nu X_2.Y_2 \cap X_1 \cap X_2$, or equivalently $\psi_2' = Y_2 \cap X_1$. Then $\mu X_1.Y_1 \cup X_1 \cup (Y_2 \cap X_1) = Y_1$ and $\psi_2'(X_1, Y_1, Y_2) = Y_1 \cap Y_2$. We end up with "$X_1 = Y_1$, $X_2 = Y_1 \cap Y_2$".

We now provide a notion of guardedness for systems of fixpoint equations. For this, we slightly extend our terminology and use "μ-guarded" (resp. "ν-guarded") as synonymous with "upward-guarded" (resp. "downward-guarded").

Definition 17 (Guarded systems of fixpoint equations) A system $X_1 \stackrel{\lambda_1}{=} \varphi_1(X_1, \ldots, X_n, \vec{Y}), \ldots, X_n \stackrel{\lambda_n}{=} \varphi_n(X_1, \ldots, X_n, \vec{Y})$ is *guarded* if one of the following two condition holds:

$$X_i \text{ is } \lambda_j\text{-guarded in } \varphi_j \text{ for all } 1 \leq i \leq j \leq n, \quad \text{or} \tag{C1}$$

$$X_i \text{ is } \lambda_i\text{-guarded in } \varphi_j \text{ for all } 1 \leq j \leq i \leq n. \tag{C2}$$

Lemma 18 *If $X_1 \stackrel{\lambda_1}{=} \varphi_1, \ldots, X_n \stackrel{\lambda_n}{=} \varphi_n$ is a guarded system of fixpoint equations, then the derived system—see Eq. (14)—is guarded too.*

Proof Being guarded, the system under consideration satisfies condition (C1) or (C2).

Assume that it satisfies (C1). We show that the derived system too satisfies (C1): Indeed, for $2 \leq i \leq j$ the occurrences of X_i in $\varphi_j(\psi_1, X_2, \ldots, X_n, \vec{Y})$ are either, so-called *previous*, occurrences in $\varphi_j(\_\_, X_2, \ldots, X_n, \vec{Y})$, or *new* ones in ψ_1. In the first case, they are λ_j-guarded as in the original system. In the second case, they occur in a term that replaces X_1 whose occurrences in $\varphi_j(X_1, \ldots)$ were λ_j-guarded by assumption, hence they are λ_j-guarded too in $\varphi_j(\psi_1, \ldots)$.

If, on the other hand, we assume that the original system satisfies condition (C2), we can show that the derived system too satisfies (C2). The reasoning is unchanged for previous occurrences of some X_i in some $\varphi_j(\psi_1, X_2, \ldots)$ when $2 \leq j \leq i \leq n$. For the new occurrences of X_i, we observe that they appear in the subterm $\psi_1 \stackrel{\text{def}}{=} \lambda_1 X_1.\varphi_1(X_1, X_2, \ldots, X_n, \ldots)$ where they are λ_i-guarded since, by assumption, they were λ_i-guarded in $\varphi_1(X_1, X_2, \ldots, X_n, \ldots)$. □

Theorem 19 *Let* $\vec{X} \stackrel{\Lambda}{=} \vec{\varphi}(\vec{X}, \vec{Y})$ *be a guarded system of fixpoint equations. Then* $[\![\Lambda \vec{X}.\vec{\varphi}]\!]$ *is a region-preserving mapping with values in* $(2^W)^{\vec{Y}}$. *Furthermore, if the region algebra is effective,* $[\![\Lambda \vec{X}.\vec{\varphi}]\!]$ *is computable.*

Proof The proof is by induction on the dimension n of the system, the base case where $n = 0$ holding vacuously.

Assume now that $n \geq 1$ and write ψ_1 for $\lambda_1 X_1.\varphi_1(X_1, \ldots, X_n, \vec{Y})$. Recall that $[\![\Lambda \vec{X}.\vec{\varphi}]\!]_{env}$ is $\langle U_1, U_2, \ldots, U_n \rangle$, where $\langle U_2, \ldots, U_n \rangle$ is the value over *env* of a derived system—see Eq. (14)—, and where U_1 is $[\![\psi_1]\!]_{env \oplus [X_2 \mapsto U_2, \ldots, X_n \mapsto U_n]}$.

We assume that $env(Y_j)$ is a region for all $Y_j \in \vec{Y}$ and consider the derived system. By Lemma 18, this system is guarded. The induction hypothesis applies and we deduce that $U_2, \ldots, U_n$ are regions. Furthermore they can be computed from *env* when the region algebra is effective.

Consider now ψ_1 and observe that, thanks to condition (C1) or (C2), X_1 is λ_1-guarded in φ_1. Hence, by Theorem 12, $[\![\psi_1]\!]$ is region-preserving, (and computable when the region algebra is effective). Thus $U_1 = [\![\psi_1]\!]_{env \oplus [X_2 \mapsto U_2, \ldots, X_n \mapsto U_n]}$ is also a (computable) region. □

Remark 20 For a system of n fixpoint equations, guardedness as defined in Definition 17 requires guardedness of X_i in φ_j for $\frac{n(n+1)}{2}$ pairs (i, j). Unfortunately, simpler conditions will not be sufficient for Theorem 19. Consider for example the following condition:

$$X_i \text{ is } \lambda_i\text{-guarded in } \varphi_i(X_1, \ldots, X_n, \vec{Y}) \quad \text{for all } 1 \leq i \leq n. \tag{C0}$$

Then the following system "$X_1 \stackrel{\mu}{=} X_2, X_2 \stackrel{\mu}{=} o(X_1)$" satisfies condition (C0) but its solution is $\langle V, V \rangle$ for $V = [\![\mu X.o(X)]\!]$. In general V is not a region.

Note that this example can be reproduced with any condition that tries to weaken (C1) or (C2) by omitting some of their (i, j) pairs.

5 Applications to the verification of well-structured counter systems

In the second part of this article, we show how the results of Sects. 3 and 4 apply to a variety of verification problems for well-structured transition systems (WSTS). This is not an exhaustive survey (and applications exist outside of verification).

This section focuses on basic verification problems and (several kinds of) counter systems since they are a simple and ubiquitous model. We consider lossy channel systems in the following sections since this latter model is better suited to the game-theoretical and/or probabilistic questions we use for illustration. This choice of applications is motivated by their prominence in the WSTS literature, but the majority of the results we present can be adapted more or less directly to other WSTS settings.

5.1 Well-structured transition systems

Recall that a *(labeled) transition system* is a structure $\mathcal{T} = (Conf, \Sigma, \rightarrow)$ where $Conf = \{\sigma, \tau, \gamma, \ldots\}$ is a set of *configurations*, or "states" of the system, where $\Sigma = \{\ell, \ell', \ldots\}$ is a set of *labels*, and $\rightarrow \subseteq Conf \times \Sigma \times Conf$ is a labeled *transition relation*. A transition $(\sigma, \ell, \sigma') \in \rightarrow$ is customarily called a *step* of $\mathcal{T}$ and denoted $\sigma \stackrel{\ell}{\rightarrow} \sigma'$, or just $\sigma \rightarrow \sigma'$ when the label ℓ is not relevant, or when one deals with unlabeled transition systems.

Definition 21 [6, 48, 55] A *well-structured transition system* is a transition system $\mathcal{T} = (Conf, \Sigma, \to, \sqsubseteq)$ enriched with a WQO $\sqsubseteq$ over $Conf$, and such that the transitions satisfy the following monotonicity property:[2] for all steps $\sigma \xrightarrow{\ell} \sigma'$ and configurations $\tau \sqsupseteq \sigma$, there exists a step $\tau \xrightarrow{\ell} \tau'$ such that $\tau' \sqsupseteq \sigma'$.

Using standard notations, we let $Pre[\ell](\gamma') \stackrel{\text{def}}{=} \{\gamma \in Conf \mid \gamma \xrightarrow{\ell} \gamma'\}$ denote the sets of 1-step predecessors of γ by some ℓ-labeled step. This extends to sets of configurations: $Pre[\ell](V) \stackrel{\text{def}}{=} \bigcup_{\gamma \in V} Pre[\ell](\gamma)$ for $V \subseteq Conf$; to sequences of labels (words in Σ^*): $Pre[\ell_1 \dots \ell_n](V) \stackrel{\text{def}}{=} Pre[\ell_1](\dots(Pre[\ell_n](V))\dots)$, with $Pre[\varepsilon](V) \stackrel{\text{def}}{=} V$; to languages $L \subseteq \Sigma^*$ with $Pre[L](V) \stackrel{\text{def}}{=} \bigcup_{w \in L} Pre[w](V)$. Finally, Pre^* and Pre^+, are shorthand for $Pre[\Sigma^*]$ and $Pre[\Sigma^+]$, and collect all the predecessors (resp., strict predecessors) of some configurations.

The dual $\widetilde{Pre}$ of Pre is defined by $\widetilde{Pre}(V) = Conf \setminus Pre(Conf \setminus V)$, or $\neg Pre(\neg V)$ in more compact notation. Thus $\sigma \in \widetilde{Pre}(V)$ iff all 1-step successors of σ are in V (this includes the case where σ is a deadlock state with no successors). Observe that, seen as unary operators on 2^{Conf}, Pre and $\widetilde{Pre}$ are monotonic. In fact, Pre is $\cup$-continuous for all transition systems, and $\cap$-continuous for finitely branching ones (symmetrically for $\widetilde{Pre}$).

5.2 Monotonic counter systems

A first example of WSTS is given by *monotonic counter systems*. We start with general Presburger counter systems before restricting to monotonic systems.

Informally, a counter is a storage location holding a natural number. Formally, a (Presburger) *counter system* is a tuple $S = (Q, \Sigma, d, \Delta)$ where $Q = \{p, q, \dots\}$ is a finite set of *locations*, Σ is the set of labels, $d \in \mathbb{N}$ is a *dimension*, and $\Delta \subseteq Q \times \Sigma \times Pres(X \cup X') \times Q$ is a finite set of *transition rules*, or shortly "rules". Here, $X = \{x_1, x_2, \dots, x_d\}$ is a set of d variables for the d counters, while $X' = \{x'_1, x'_2, \dots, x'_d\}$ are primed copies, and $Pres(X \cup X')$ is the set of Presburger formulae with free variables in $X \cup X'$. The components of a rule $\delta = (p, \ell, u, q)$ are a start- and an end-location p, q, a label ℓ, and an *update* $u(X, X')$, i.e., a Presburger formula describing how the current values of the counters change when firing δ.

The operational semantics is as expected. A *configuration* of $S = (Q, \Sigma, d, \Delta)$ is a pair $\sigma = (p, \mathbf{a})$ of a *current location* $p \in Q$ and a *current valuation* $\mathbf{a} \in \mathbb{N}^d$ of the counters. Transitions between configurations are obtained from the rules: there is a transition $(p, \mathbf{a}) \xrightarrow{\ell} (q, \mathbf{b})$ if Δ contains a rule $\delta = (p, \ell, u, q)$ such that u is satisfied when the values $\mathbf{a}$ and $\mathbf{b}$ are assigned to X and X', denoted $\models u(\mathbf{a}, \mathbf{b})$.

Example 22 (A simple counter system) Figure 1 depicts a simple (unlabeled) counter system with $d = 2$ and $Q = \{r, s\}$. Its behavior in any given configuration is completely deterministic. Here is a run starting from $(r, 0, 0)$:

$$(r,0,0) \to (s,0,0) \to (r,1,0) \to (r,0,1) \to (s,0,1) \to (s,1,0) \to (r,1,1)$$
$$\to (r,0,2) \to (s,0,2) \to (s,1,1) \to (s,2,0) \to (r,2,1) \to \cdots .$$

It appears that all possible configurations will be visited exactly once.

[2] Called "strong compatibility" in [48]. There exist alternative definitions of well-structured transition systems based on weaker notions of compatibility.

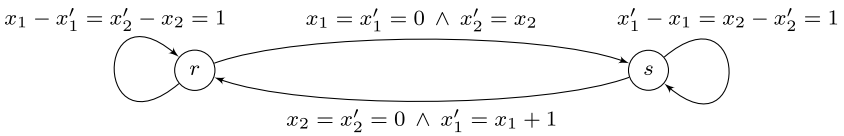

Fig. 1 A simple counter system

Example 23 (Minsky Machines and Vector Addition Systems) Using arbitrary Presburger formulae for updates is expressive and versatile. It generalizes many classical models like Minsky Machines or Vector Addition Systems, etc., where only updates of a specific form are allowed. For Vector Addition Systems, the updates have the form

$$u_{(\mathbf{v}^-,\mathbf{v}^+)}\big(X, X'\big) \stackrel{\text{def}}{\Leftrightarrow} \bigwedge_{j=1}^{d}\big(x_j \geq \mathbf{v}^-[j] \wedge x_j' = x_j - \mathbf{v}^-[j] + \mathbf{v}^+[j]\big) \tag{18}$$

for some $\mathbf{v}^+, \mathbf{v}^- \in \mathbb{N}^d$, or equivalently, when seeing X and X' as vectors,

$$u_{(\mathbf{v}^-,\mathbf{v}^+)}\big(X, X'\big) \stackrel{\text{def}}{\Leftrightarrow} X \geq \mathbf{v}^- \wedge X' = X - \mathbf{v}^- + \mathbf{v}^+. \tag{19}$$

For Minsky Machines, the updates are *zero tests*, written "`if` $x_i = 0$ `then` ..." in programming notation, *incrementations* "x_i := $x_i + 1$", and decrementations "if $x_i > 0$ `then` x_i := $x_i - 1$". This imperative notation is more compact but less explicit than their Presburger formulation:

$$u_{\text{zero?}(x_i)}\big(X, X'\big) \stackrel{\text{def}}{\Leftrightarrow} x_i = 0 \wedge \bigwedge_{j=1}^{d} x_j = x_j', \tag{20}$$

$$u_{\text{incr}(x_i)}\big(X, X'\big) \stackrel{\text{def}}{\Leftrightarrow} x_i' = x_i + 1 \wedge \bigwedge_{\substack{j=1 \\ j\neq i}}^{d} x_j = x_j', \tag{21}$$

$$u_{\text{decr}(x_i)}\big(X, X'\big) \stackrel{\text{def}}{\Leftrightarrow} x_i = x_i' + 1 \wedge \bigwedge_{\substack{j=1 \\ j\neq i}}^{d} x_j = x_j'. \tag{22}$$

Now to monotonicity. For this, we order the set $Conf_S \stackrel{\text{def}}{=} Q \times \mathbb{N}^d$ of configurations of S with

$$(p, \mathbf{a}) \leq (q, \mathbf{b}) \stackrel{\text{def}}{\Leftrightarrow} p = q \wedge \mathbf{a} \leq \mathbf{b}. \tag{23}$$

This extends the product ordering on $\mathbb{N}^d$ (from Example 6) and turns $Conf_S$ into a WQO since $(\mathbb{N}^d, \leq)$ is a WQO and Q is finite. For counter systems, a natural choice for symbolic verification purposes is to use the algebra $\mathcal{R}_{\mathbf{SL}}(S)$ over $Conf_S$ of semilinear,[3] or equivalently Presburger-definable, regions of S.

[3]Semilinear subsets of $Conf_S = Q \times \mathbb{N}^d$ are all sets of the form $\bigcup_{q\in Q}\{q\} \times R_q$ where each $R_q \subseteq \mathbb{N}^d$ is semilinear. Equivalently, they can be seen as semilinear subsets of $\mathbb{N}^{d+1}$, or more precisely of $\{0, 1, 2, \ldots, |Q| - 1\} \times \mathbb{N}^d$, by identifying Q with its cardinal.

Definition 24 A counter system is *monotonic* if the associated transition system $\mathcal{T}_S \stackrel{\text{def}}{=} (Conf_S, \Sigma, \to, \leq)$ is well structured.

Observe that the question whether $S = (Q, \Sigma, d, \Delta)$ is monotonic can be expressed by a Presburger formula,

$$\bigwedge_{\substack{p,q\in Q\\ \ell\in\Sigma}} \left[\left(Y \geq X \wedge \bigvee_{(p,\ell,u,q)\in\Delta} u(X,X')\right) \Rightarrow \left(\exists Y' : Y' \geq X' \wedge \bigvee_{(p,\ell,u',q)\in\Delta} u'(Y,Y')\right)\right], \tag{24}$$

hence is decidable.

Example 25 (Some classes of monotonic counter systems)

1. *Vector addition systems*, or equivalently *Petri nets*, are monotonic as can be checked on Eq. (18).
2. Many *extensions of Petri nets* allow richer sets of updates (on the same set of configurations) and retain monotonicity. For example, *Post-self-modifying nets* [79], reset/transfer nets [38] and broadcast protocols [40], or the more general *affine nets* [47]. These are all special cases of Presburger counter systems since their updates can be defined with Presburger formulae.
3. *Lossy counter machines* are monotonic: see Sect. 5.4.

Remark 26 Famously, *Minsky machines are not monotonic* because of their zero tests.

We now consider the monotonic region algebra $\mathcal{R}_{\mathbf{SL}}(S)$. The signature includes union, intersection, $C_\uparrow$ and $K_\uparrow$, the $Pre[\ell]$ operators (all monotonic operators), and some constants. In terms of this region algebra, the defining property of monotonic counter systems entails

$$Pre[\ell](C_\uparrow R) = C_\uparrow\big(Pre[\ell](C_\uparrow R)\big). \tag{25}$$

We abuse terminology and say that a constant c is upward-closed if its interpretation $[\![c]\!]$ is.

Proposition 27 *Let φ be a closed L_μ term that uses only Pre, union, intersection, upward-closure, fixpoints and upward-closed constants. Then, for monotonic counter systems, $[\![\varphi]\!]$ is upward-closed.*

Proof We show that for any subformula ψ of φ, $[\![\psi(X_1,\ldots)]\!]_{env}$ is upward-closed when $env(X_1),\ldots$ are upward-closed. The result will apply to φ without any restriction since it has no free variable and $[\![\varphi]\!]$ does not depend on env.

The proof is by structural induction on ψ. We consider all cases in turn. If ψ is a constant c, $[\![c]\!]$ is upward-closed by assumption. If ψ is a variable X, $[\![X]\!]_{env} = env(X)$ is upward-closed by assumption. If ψ is some $\psi_1 \cap \psi_2$ or $\psi_1 \cup \psi_2$, we know by induction hypothesis that $[\![\psi_1]\!]_{env}$ and $[\![\psi_2]\!]_{env}$ are upward-closed, hence $[\![\psi]\!]_{env}$ too since the union or the intersection of two upward-closed sets is upward-closed. If ψ is some $Pre[\ell](\psi')$ we rely on the fact that $[\![\psi']\!]_{env}$ is upward-closed (induction hypothesis), so that $[\![\psi]\!]_{env} = Pre[\ell]([\![\psi']\!]_{env}) = Pre[\ell](C_\uparrow[\![\psi']\!]_{env})$ is upward-closed by Eq. (25). If ψ is some $\mu X.\psi'$, we first prove, by induction over i, that all the approximants $(M_i)_{i\in\mathbb{N}}$ of $[\![\mu X.\psi']\!]_{env}$, are upward-closed, relying on the induction hypothesis on ψ' to prove that

$M_{i+1} = [\![\psi']\!]_{env \oplus [X \mapsto M_i]}$ is upward-closed. Now, by Fact 1, the sequence of approximants stabilize in finite time since they are increasing and upward-closed. Thus $[\![\psi]\!]_{env}$ coincides with some M_k, and it is upward-closed. The case where ψ is some $\nu X.\psi'$ is similar. □

Let B($\exists$U, $\exists$X, $\wedge$, $\vee$) denote the fragment of CTL (the well-known temporal logic, see [39] for notations and definitions) where only existential path quantification is allowed. Write $S, \sigma \models \varphi$ when configuration σ of S satisfies temporal formula φ, and $Sat_S(\varphi)$ for $\{\sigma \in Conf_S \mid S, \sigma \models \varphi\}$. *Model checking* is the problem of deciding whether $S, \sigma \models \varphi$ for given S, σ and φ.

Theorem 28 (Model-checking monotonic counter systems) *For a monotonic counter system S, and for a* B($\exists$U, $\exists$X, $\wedge$, $\vee$) *formula φ where atomic propositions are upward-closed, the set $Sat_S(\varphi)$ is an upward-closed set (hence a region in $\mathcal{R}_{\mathbf{SL}}(S)$) that can be computed from S and φ.*

Proof By translating CTL as L_μ terms in the classical way [30], e.g., with

$$\exists[\varphi_1 \mathsf{U} \varphi_2] \text{ is CTL notation for } \mu X.\varphi_2 \cup \varphi_1 \cap Pre(X), \tag{26}$$

we can apply Proposition 27 and deduce that all $Sat_S(\varphi)$ are upward-closed. Therefore, replacing any $Pre(\ldots)$ by $Pre(C_\uparrow(\ldots))$ in the resulting L_μ terms does not change their interpretation.

Now, since B($\exists$U, $\exists$X, $\wedge$, $\vee$) only uses least fixpoints, and all such fixpoints have the bound variable under the scope of *Pre* as exemplified in Eq. (26), this gives us terms that are *upward-guarded*. Thus they can be evaluated as a consequence of Theorem 12. □

Thus, model checking B($\exists$U, $\exists$X, $\wedge$, $\vee$) is decidable under the above assumptions. This CTL fragment allows combining and nesting the constrained reachability properties that are central to the verification of safety properties.

Remark 29 Theorem 28 cannot be extended to the whole of CTL. For example, the set $Sat_S(\forall\Diamond r)$ of all configurations from which one will inevitably reach a given control state $r \in Q$ cannot be computed from S, even when S is a lossy counter system [77].

5.3 Regular simulation for monotonic counter systems

In process algebra, so-called "*regular*" equivalences and preorders are behavioral relations between an arbitrary labeled transition system and a *finite-state* one (which is usually taken as a specification of the other process) [58, 65].

We first consider *regular simulation*, i.e., the special case where one considers Milner's classic simulation preorder.[4]

Consider an arbitrary counter system $S = (Q, \Sigma, d, \Delta)$ and assume that $\mathcal{F} = (F, \Sigma, \rightarrow_{\mathcal{F}})$ is a *finite-state* transition system, i.e., $F = \{f_1, \ldots, f_n\}$ for some n. The *simulation relation*

[4]Regular simulation is known to be decidable for well-structured systems [6]. By contrast, outside of the regular equivalence framework, all sensible behavioral relations between configurations of monotonic counter systems are undecidable [56, 75].

between S and $\mathcal{F}$, is the largest relation $\preceq \subseteq Conf_S \times F$ that satisfies the following transfer property (see, e.g., [50]):

$$\sigma \preceq f \quad \Longrightarrow \quad \text{for all } \sigma \xrightarrow{\ell} \sigma' \text{ there exists } f \xrightarrow{\ell}_{\mathcal{F}} f' \text{ s.t. } \sigma' \preceq f'. \tag{27}$$

For $f \in F$, let $X_f \stackrel{\text{def}}{=} \{\sigma \in Conf_S \mid \sigma \preceq f\}$.

Theorem 30 (Decidability for regular simulation) *If S is a monotonic counter system, the sets $(X_f)_{f\in F}$ are downward-closed and can be computed effectively.*

Proof Since F is finite, the coinductive definition of $\preceq$ translates into a finite system of fixpoint equations for the X_f sets. Formally, Eq. (27) rewrites as

$$X_{f_1} \stackrel{\nu}{=} \bigcap_{\ell\in\Sigma} \widetilde{Pre}[\ell]\Big(\bigcup_{f_1 \xrightarrow{\ell}_{\mathcal{F}} f'} X_{f'}\Big), \quad \ldots, \quad X_{f_n} \stackrel{\nu}{=} \bigcap_{\ell\in\Sigma} \widetilde{Pre}[\ell]\Big(\bigcup_{f_n \xrightarrow{\ell}_{\mathcal{F}} f'} X_{f'}\Big). \tag{28}$$

By duality, the complementary sets $Y_f \stackrel{\text{def}}{=} Conf_S \setminus X_f$ are given by the following system of fixpoint equations:

$$Y_{f_1} \stackrel{\mu}{=} \bigcup_{\ell\in\Sigma} Pre[\ell]\Big(\bigcap_{f_1 \xrightarrow{\ell}_{\mathcal{F}} f'} Y_{f'}\Big), \quad \ldots, \quad Y_{f_n} \stackrel{\mu}{=} \bigcup_{\ell\in\Sigma} Pre[\ell]\Big(\bigcap_{f_n \xrightarrow{\ell}_{\mathcal{F}} f'} Y_{f'}\Big). \tag{29}$$

With Proposition 27 we know that the solutions of Eq. (29) are upward-closed when S is monotonic. Hence it is equivalent to define the $(Y_f)_{f\in F}$ with:

$$Y_{f_1} \stackrel{\mu}{=} \bigcup_{\ell\in\Sigma} Pre[\ell]\Big(C_\uparrow \bigcap_{f_1 \xrightarrow{\ell}_{\mathcal{F}} f'} Y_{f'}\Big), \quad \ldots, \quad Y_{f_n} \stackrel{\mu}{=} \bigcup_{\ell\in\Sigma} Pre[\ell]\Big(C_\uparrow \bigcap_{f_n \xrightarrow{\ell}_{\mathcal{F}} f'} Y_{f'}\Big), \tag{30}$$

which is guarded. Theorem 19 now applies and gives us computability. □

It is easy to extend Theorem 30 to (most variants of) *weak simulation*, where a special label $\tau \in \Sigma$ is a silent internal action [49]. This means replacing the existential quantification "$\exists\, f \xrightarrow{\ell}_{\mathcal{F}} f'$" in Eq. (27) with, e.g., "$\exists\, f \xrightarrow{\tau^*\ell\tau^*}_{\mathcal{F}} f'$" and leads to a variant of Eq. (28) that is still guarded.

Remark 31 (Simulation of $\mathcal{F}$ by S) One cannot compute the simulation relation in the other direction, i.e., of $\mathcal{F}$ by S. Let $X'_f = \{\sigma \mid f \preceq \sigma\}$. The definition of simulation leads to a system of fixpoint equations for the $(X'_f)_{f\in F}$ that is not guarded. From this we deduce that each X'_q is (semilinear and) upward-closed, using Proposition 27. However, X'_f is not computable[5] from S and $\mathcal{F}$ since if $\mathcal{F}$ is just a single loop "$f_1 \to f_1$", X'_{f_1} is exactly the set of configurations from which S has an infinite run, a set that cannot be computed even for lossy counter machines [77].

[5]However, it is decidable whether $f \preceq \sigma$ (given σ) when S is finitely branching [6].

5.4 Lossy counter machines

Lossy counter machines are a weak version of counter machines that is well structured by construction. It is convenient to define them as standard counter machines with a new operational semantics that defines their unreliability [68, 77]. Formally, given a Presburger counter machine S, its lossy variant S_{lossy} is obtained by replacing any update $u(X, X')$ in the rules of S by a modified, "lossy", version

$$u_{\text{lossy}}\big(X, X'\big) \overset{\text{def}}{\Leftrightarrow} \exists Y, Y' : X \geq Y \wedge u\big(Y, Y'\big) \wedge Y' \geq X'. \tag{31}$$

The behavior of the resulting S_{lossy} is the behavior of S except that one now assumes that the counters are unreliable and can decrease nondeterministically before and after steps.

Equation (31) entails that S_{lossy} is a monotonic system. In fact, lossy counter machines satisfy a stronger property:

$$\begin{aligned} Pre[\ell](R) &= Pre[\ell](C_{\uparrow}R) = C_{\uparrow}\big(Pre[\ell](C_{\uparrow}R)\big), \\ \widetilde{Pre}[\ell](R) &= \widetilde{Pre}[\ell](K_{\downarrow}R) = K_{\downarrow}\big(\widetilde{Pre}[\ell](K_{\downarrow}R)\big). \end{aligned} \tag{32}$$

This leads to a notable strengthening of Theorem 28: one can handle negation!

Theorem 32 *Model checking* $\mathsf{B}(\exists\mathsf{U}, \exists\mathsf{X}, \wedge, \vee, \neg)$ *is decidable for lossy counter systems (assuming that atomic propositions are semilinear). Furthermore, for every formula* φ *of this fragment of* CTL*, the set* $Sat_S(\varphi)$ *is semilinear and computable from* S *and* φ*.*

Proof [Sketch] Using Eq. (32) the μ-calculus definitions of $\mathsf{B}(\exists\mathsf{U}, \exists\mathsf{X}, \wedge, \vee, \neg)$ formulae directly lead to guarded terms. □

We are also in a position to deal with regular bisimulation. Write $Z_f \overset{\text{def}}{=} \{\sigma \in Conf_S \mid \sigma \sim f\}$ for the set of configurations of S that are bisimilar with a state f of $\mathcal{F}$. The natural coinductive definition of bisimulation does not lead to a guarded system for the Z_f's. However, for regular bisimulation, an alternative characterization leads to an inductive definition as we now explain (see also [58, 65]). Recall the definition of the finite-depth approximants $\sim_0 \supseteq \sim_1 \supseteq \sim_2 \supseteq \cdots$ of bisimulation:

$$\sigma \sim_{n+1} f \overset{\text{def}}{\Leftrightarrow} \begin{cases} \forall \sigma \xrightarrow{\ell} \sigma' : \exists f \xrightarrow{\ell}_{\mathcal{F}} f' : \sigma' \sim_n f', \text{ and} \\ \forall f \xrightarrow{\ell}_{\mathcal{F}} f' : \exists \sigma \xrightarrow{\ell} \sigma' : \sigma' \sim_n f', \end{cases} \tag{33}$$

where $\sigma \sim_0 f$ holds for all σ and f. Then, writing N for $|F|$, $\sigma \sim f$ iff

$$\sigma \sim_N f \text{ and for all } \sigma \xrightarrow{*} \sigma' \text{ there is a } f' \in F \text{ s.t. } \sigma' \sim_N f'. \tag{34}$$

Using Eq. (32), the above characterization yields a guarded system for $(Z_q)_{q \in F}$:

$$Z_f = Z_f^N \cap Z_F, \qquad Z_F \overset{v}{=} \left(\bigcup_{f' \in F} Z_{f'}^N \cap \bigcap_{\ell \in \Sigma} \widetilde{Pre}[\ell](K_{\downarrow} Z_F) \right),$$

$$Z_f^0 = Conf_S, \qquad Z_f^{n+1} = \bigcap_{\ell \in \Sigma} \left(\bigcap_{f \xrightarrow{\ell}_{\mathcal{F}} f'} Pre[\ell](Z_{f'}^n) \cap \widetilde{Pre}[\ell] \left(\bigcup_{f \xrightarrow{\ell}_{\mathcal{F}} f'} Z_{f'}^n \right) \right).$$

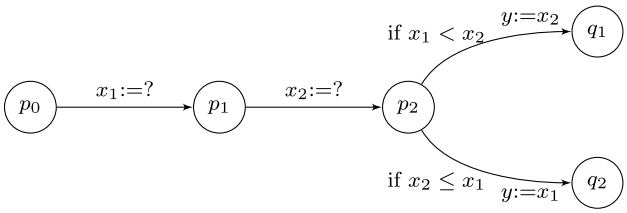

Fig. 2 An integral relational automaton

Corollary 33 (Decidability for regular bisimulation) *For lossy counter systems, the sets* $(Z_f)_{f\in F}$ *are semilinear and can be computed effectively.*

Remark 34 The above result does not extend to all monotonic systems. For example, even in the simple case of VASS's for which regular bisimulation is decidable [57], the sets $(Z_f)_{f\in F}$ are in general not semilinear.

5.5 Integral relational automata

Integral relational automata (IRA) [32] are counter systems where updates are restricted to assignments guarded by comparisons. Using an imperative notation, the assignments can have the form $x_i := c$ for a counter x_i and a constant $c \in \mathbb{N}$, $x_i := x_j$ for two counters, and the special $x_i := ?$ that assigns nondeterministically any natural number (and yields infinitely-branching transition systems). These updates are guarded by arbitrary Boolean combinations of simple tests of the form $x_i < c$ (comparing a counter and a constant) and $x_i < x_j - c$ (comparing two counters, with gap-order constraints allowed). Since these updates are Presburger-definable, IRA's are a special case of counter systems.

Example 35 The IRA depicted on Fig. 2 nondeterministically picks two arbitrary values for x_1 and x_2, compares them and stores the largest one in y.

IRA's are not monotonic in the sense of Definition 24, however they are well structured transition systems when one assumes a different ordering on configurations. Formally, for $\mathbf{a} = (a_1, \ldots, a_d)$ and $\mathbf{b} = (b_1, \ldots, b_d)$ in $\mathbb{N}^d$ or in $\mathbb{Z}^d$, we say that $\mathbf{a}$ is *sparser* than $\mathbf{b}$, written $\mathbf{a} \leq_{sp} \mathbf{b}$, when

$$\text{for all } 1 \leq i, j \leq d\text{:} \quad (a_i \leq a_j \text{ iff } b_i \leq b_j) \text{ and } |a_i - a_j| \leq |b_i - b_j|. \tag{35}$$

In other words, the ordering between any two elements of $\mathbf{a}$ is respected by the corresponding elements in $\mathbf{b}$, and their relative distance is not decreased.

Fact 36 $(\mathbb{N}^d, \leq_{sp})$ *and* $(\mathbb{Z}^d, \leq_{sp})$ *are WQO's.*

Indeed there are only a finite number $F_d \approx \frac{d!}{2(\log 2)^{d+1}}$ of total preorders of d elements,[6] and only $d^{\frac{d-1}{2}}$ distinct differences between the d elements. Thus there exists an order-embedding from $(\mathbb{Z}^d, \leq_{sp})$ to $(F_d \times \mathbb{N}^{d^{\frac{d-1}{2}}}, \leq)$. Furthermore, the $d^{\frac{d-1}{2}}$ differences can

[6]The F_d's are known as the "Fubini numbers", or the "ordered Bell numbers", see A000670 in the Encyclopedia of Integer Sequences.

be generated with only the $d-1$ differences between immediate neighbors since if, e.g., $x < y < z$ then $|x-z| = |x-y| + |y-z|$. Thus $(\mathbb{Z}^d, \leq_{sp})$ can even be embedded into $(F_d \times \mathbb{N}^{d-1}, \leq)$. Since the latter is a WQO (by Dickson's Lemma), the former is too.

When $C = \{c_1, \ldots, c_k\}$ is a finite subset of $\mathbb{Z}$ (with $c_1 < c_2 < \cdots < c_k$), we let

$$\begin{aligned} \mathbf{a} \leq_{sp}^{C} \mathbf{b} &\stackrel{\text{def}}{\Leftrightarrow} (a_1, \ldots, a_d, c_1, \ldots, c_k) \leq_{sp} (b_1, \ldots, b_d, c_1, \ldots, c_k) \\ &\Leftrightarrow (a_1, \ldots, a_d, c_1, c_k) \leq_{sp} (b_1, \ldots, b_d, c_1, c_k). \end{aligned} \tag{36}$$

Then, for any C, $(\mathbb{Z}^d, \leq_{sp}^{C})$ is a WQO that refines $(\mathbb{Z}^d, \leq_{sp})$.

If we now let C_S be the (finite) set of all constants that appear in a guard or an assignment of S and order $Conf_S$ with $(p, \mathbf{a}) \sqsubseteq_S (q, \mathbf{b}) \stackrel{\text{def}}{\Leftrightarrow} p = q \wedge \mathbf{a} \leq_{sp}^{C_S} \mathbf{b}$, we obtain a WQO w.r.t. which the IRA S is well structured [6, 32].

Now to verification: IRA's are counter systems, so we can use the algebra of semilinear regions for symbolic verification, with *Pre* and *Post* being effective and region-preserving. Configurations are well quasi-ordered with $\leq_{sp}^{C_S}$: since this WQO is Presburger-definable (witness Eqs. (35) and (36)), the associated closure and interior operators are effective and region-preserving. Finally, all the machinery described above for the symbolic verification of monotonic counter machines still apply *mutatis mutandis*.

Theorem 37 *Regular simulation and model-checking* $\mathsf{B}(\exists\mathsf{U}, \exists\mathsf{X}, \wedge, \vee)$ *where atomic propositions are upward-closed (w.r.t.* $\sqsubseteq_S$*) are decidable for IRA's.*[7]

Furthermore, for any $\mathsf{B}(\exists\mathsf{U}, \exists\mathsf{X}, \wedge, \vee)$ *formula* φ*, the set* $Sat_S(\varphi)$ *is upward-closed and computable, and for any state* q *of a finite* $\mathcal{F}$*, the set* $X_q = \{\sigma \mid \sigma \preceq q\}$ *is downward-closed and computable.*

5.6 Incrementation errors

Counter automata with incrementation errors, called ICA's for short, have recently been introduced in connection with temporal logics and data logics [37, 44]. Seen as counter systems, they are not monotonic, but "co-monotonic", i.e, they satisfy an equation similar to Eq. (24) where $Y \geq X$ and $Y' \geq X'$ are replaced with $Y \leq X$ and $Y' \leq X'$. Equivalently, the transition system obtained by reversing the direction of steps (running the system backwards) is well structured. Thus, instead of Eq. (32), one relies on $Post[\ell](R) = C_{\uparrow}(Post[\ell](C_{\uparrow}R))$ to prove, e.g., that for ICA's $Post^*(R)$ is a region and can be computed effectively.

6 Lossy channel systems

Lossy channel systems (LCS's) are another classic model for which WSTS theory provides many positive results [9, 31]. In this section, we first recall the necessary notations, definitions and classical results before considering the verification of game-theoretical and probabilistic properties in Sects. 7–9.

For simplicity we consider unlabeled systems. A channel system is a tuple $S = (Q, \mathsf{C}, \mathsf{M}, \Delta)$ consisting of a finite set $Q = \{q, q', \ldots\}$ of *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*

[7] In the case of IRA's, the optimal complexity is obtained with algorithms that are not WSTS-based [29].

rules. Each transition rule has the form (q, op, q'), written $q \xrightarrow{op} q'$, where op is an *operation*: $c!m$ (sending message $m \in \mathsf{M}$ along channel $c \imath n \mathsf{C}$), $c?m$ (receiving message m from channel c), or $\surd$ (an internal action to some process, no I/O-operation).

With a channel system S, we associate two operational semantics. We start with the "reliable" semantics: A *configuration* of S is a pair (q, w) where q is a location of S and $w : \mathsf{C} \to \mathsf{M}^*$ is a mapping, called the *channel valuation*, that describes the channel contents. Thus $Conf_S \stackrel{\text{def}}{=} Q \times \mathsf{M}^{*\mathsf{C}}$. When $\mathsf{C} = \{c_1, \ldots, c_d\}$, we often use M^{*d} instead of $\mathsf{M}^{*\mathsf{C}}$, and write $(q, u_1, \ldots, u_d)$ instead of (q, w) (when $u_i = w(c_i)$ for $i = 1, \ldots, d$). Reliable, also called "perfect", steps are as expected. Formally, the effect of an operation op on the contents w, denoted $op(w)$, is the valuation w' such that:[8]

$$\begin{aligned}
&\text{if } op = c!m\text{:} \quad && w'(c) = w(c).m \text{ and } w'\big(c'\big) = w\big(c'\big) \text{ for all } c' \neq c,\\
&\text{if } op = c?m\text{:} \quad && m.w'(c) = w(c) \text{ and } w'\big(c'\big) = w\big(c'\big) \text{ for all } c' \neq c,\\
&\text{if } op = \surd\text{:} \quad && w'(c) = w(c) \text{ for all } c.
\end{aligned}$$

Then the perfect steps of S, denoted $\sigma \to_{\text{perf}} \sigma'$, are all $(q, w) \to_{\text{perf}} (q', w')$ such that there is a rule $q \xrightarrow{op} q'$ in Δ and $w' = op(w)$, in which case we say that the rule is *enabled*. We write $En(\sigma) \subseteq \Delta$ for the set of rules enabled in σ.

Now to unreliability. Losing messages is formalized via the subword ordering, extended from M^* to $Conf_S$ in the natural way: $(q, w) \sqsubseteq (q', w')$ if $q = q'$ and $w(c) \sqsubseteq w'(c)$ for all channels $c \in \mathsf{C}$. Then the semantics of S consists of so-called "lossy" steps defined with

$$\sigma \to \tau \stackrel{\text{def}}{\Leftrightarrow} \sigma \to_{\text{perf}} \tau' \wedge \tau' \sqsupseteq \tau \text{ for some } \tau'. \tag{37}$$

In plain words, a step in the LCS can be seen as a perfect step followed by arbitrary message losses.

Remark 38 (On defining lossy steps) With Eq. (37), we opt for a semantics where message losses occur anywhere in the channels, right after a perfect step. In the literature, one often considers more liberal definitions with arbitrary losses before and after a step [9], or more restrictive definitions where messages can only be lost during the steps that (try to) send them to a channel [33] or when they are in position to be read at the head of a channel [45].

There is usually no essential semantical difference between these definitions that package the same atomic events into different single "steps". The liberal definition is often technically smoother because $(Conf_S, \to_{\text{liberal}}, \sqsubseteq)$ is a WSTS. However, losing messages at the start of a step is unnatural in situations where several adversarial processes compete, e.g., in the game-theoretical or probabilistic settings that we explore in Sects. 7–9. Hence our choice of Eq. (37) for the definition of lossy steps.

One consequence of using Eq. (37) is that when $\sigma \sqsubseteq \sigma'$, the transition rules enabled in σ are not necessarily enabled in σ'. This motivates the introduction of the following relation:

$$\sigma \preceq_S \sigma' \stackrel{\text{def}}{\Leftrightarrow} \sigma \sqsubseteq \sigma' \wedge En(\sigma) \subseteq En\big(\sigma'\big). \tag{38}$$

Lemma 39 *$(Conf_S, \preceq_S)$ is a WQO and $\mathcal{T}_S \stackrel{\text{def}}{=} (Conf_S, \to, \preceq_S)$ is a WSTS.*

[8] For "$c?m$" operations, $op(w)$ is only defined if the contents of channel c starts with m.

Proof $\preceq_S$ is a WQO as the intersection of two WQO's. For monotonicity of steps, assume $\sigma \preceq_S \sigma'$ and $\sigma \to \tau$, i.e., $\sigma \to_{\text{perf}} \tau_1 \sqsupseteq \tau$ where the perfect step is obtained by firing rule $\delta \in \Delta$. From $En(\sigma) \subseteq En(\sigma')$ we conclude that there is a step $\sigma' \to_{\text{perf}} \tau_1'$ using δ. From $\sigma' \sqsupseteq \sigma$ we get $\tau_1' \sqsupseteq \tau_1$ hence $\tau_1' \sqsupseteq \tau$. Thus by letting $\tau' = \tau$, we have a step $\sigma' \to \tau'$ with $\tau \preceq_S \tau'$. □

6.1 An effective region algebra for LCS's

We adapt the algebra of regular regions (Example 4) to LCS's. Formally, for an LCS S, a *recognizable region* is any $R \subseteq Conf_S$ that can be written under the form $R = \bigcup_{i \in I} \{q_i\} \times L_i^1 \times \cdots \times L_i^d$ where I is a *finite* index set, where $d = |\mathsf{C}|$ is the number of channels in S, and where, for $i \in I$, q_i is some location $\in Q$, and each L_i^j for $j = 1, \ldots, d$ is a regular language $\in \mathbf{Reg}(\mathsf{M})$. We denote with $\mathcal{R}_{\mathbf{Rec}}(S)$ the monotonic region algebra that contains exactly these recognizable regions. It is closed under Boolean operations.

Obviously, for any transition rule $\delta \in \Delta$, the set of configurations in which δ is enabled is a recognizable region. Thus, from the regularity of language closures w.r.t. the subword ordering (see Example 5), we deduce the recognizability of the closures w.r.t. $\preceq_S$, denoted $C_{\uparrow \preceq_S} R$ and $C_{\downarrow \preceq_S} R$, or more simply $C_\uparrow R$ and $C_\downarrow R$ when S is understood, of any $R \subseteq Conf_S$. Consequently, the interiors w.r.t. $\preceq_S$, simply denoted $K_\uparrow R$ and $K_\downarrow R$, are also recognizable regions.

The set of operators we consider on regions includes $C_\uparrow$, $C_\downarrow$, $K_\uparrow$, $K_\downarrow$ (as required by the definition), union and intersection, Pre and $\widetilde{Pre}$. These operators are monotonic, region-preserving and effective.

Let us show that, even with our specific operational semantics, $Pre(R)$ and $\widetilde{Pre}(R)$ are regions. First let $Pre_{\text{perf}}(\ldots)$ denote the set of 1-step predecessors by perfect steps. It is easy to see that, when R is recognizable, $Pre_{\text{perf}}(R)$ is recognizable too: it is obtained from R by simple regularity-preserving operations like replacing the contents L_i of channel c_i with $m.L_i$ when accounting for operation $c_i?m$, and with the right-residual $(L_i)m^{-1} \stackrel{\text{def}}{=} \{x \mid xm \in L_i\}$ when accounting for $c_i!m$. We conclude with $Pre(R) = Pre_{\text{perf}}(C_\uparrow R)$ from Eq. (37). Since $C_\uparrow R$ is recognizable, $Pre(R)$ is recognizable for any R.

Finally, $\mathcal{R}_{\mathbf{Rec}}(S)$ is an effective region algebra where recognizable regions and the above-mentioned operations, including Pre and $\widetilde{Pre}$, can be handled algorithmically, e.g., using automata-based representations like QDD's [21].

Remark 40 (Lossy channel systems with guards) Guards are additional conditions that restrict the firability of rules and are useful in many situations, e.g., for modeling priorities between rules. They can be simple conditions like the emptiness of a channel, or more complex ones like the (non-)occurrence tests used in [27, 28].

It turns out that our approach can handle all guards defined by recognizable regions. If we write $q \xrightarrow{G:op} q'$ for rule $q \xrightarrow{op} q'$ guarded by G, a formal definition of the semantics would give $Pre[q \xrightarrow{G:op} q'](R) = G \cap Pre[q \xrightarrow{op} q'](R)$ which is easily accommodated by our region algebra. We will not pursue this direction in more depth, and just remark that all our results on lossy channel systems extend directly to a setting with recognizable guards.

6.2 Verification of lossy channel systems

Theorem 12 has numerous applications for LCS's, as we describe now and in the next three sections. In order to ensure guardedness of terms, we rely on the following two key equalities

$$Pre(R) = Pre(C_\uparrow R), \qquad \widetilde{Pre}(R) = \widetilde{Pre}(K_\downarrow R). \tag{39}$$

These are direct consequences of Eqs. (37) and (38) (recall that $C_\uparrow$ and $K_\downarrow$ refer to $\preceq_S$). Using Eq. (39), $Pre^*(R)$ can be expressed by a guarded term:

$$Pre^*(R) = \mu X.R \cup Pre(X) = \mu X.R \cup Pre(C_\uparrow X). \tag{40}$$

Corollary 41 *For recognizable $R \subseteq Conf$, $Pre^*(R)$ is recognizable and effectively computable.*

More generally, model checking $\mathsf{B}(\exists\mathsf{U}, \exists\mathsf{X}, \wedge, \vee, \neg)$ is decidable for lossy channel systems (assuming that atomic propositions are recognizable), as are regular simulation and bisimulation: everything we illustrated for lossy counter machines in Sect. 5.4 extends to LCS's.

Remark 42 (Beyond safety) Inevitability properties, and recurrent reachability can be stated in L_μ [30]. With temporal logic notation, this yields

$$Sat(\forall\Diamond R) = \mu X.R \cup \big(Pre(Conf) \cap \widetilde{Pre}(X)\big) \qquad \text{(Inev)},$$
$$Sat(\exists\Box\Diamond R) = \nu X.\mu Y.\big(\big[R \cup Pre(Y)\big] \cap Pre(X)\big) \qquad \text{(RecReach)}.$$

These two L_μ terms are not guarded and Eq. (39) is of no help here. This is to be expected: first, $\sigma \models \exists\Box\Diamond R$ is undecidable for lossy channel systems or even lossy counter systems [8, 77]; second, and while $\sigma \models \forall\Diamond R$ is decidable for lossy channel systems, the map $R \mapsto Sat(\forall\Diamond R)$ is not computable [68, 77].

7 Solving games on lossy channel systems

Theorem 12 is useful when solving games on LCS's, as we show in this section and in Sect. 9 for stochastic games.

From now on, all the situations we consider have game-theoretical aspects and we shall always explicit who has control over message losses: it can be one the players in this section, or the environment in the last two sections with probabilistic message losses. Therefore, Pre and $\widetilde{Pre}$ will always refer to perfect steps, i.e., they are what was written Pre_{perf} and $\widetilde{Pre}_{\text{perf}}$ earlier. For the same reasons, we order configurations with $\sqsubseteq$ rather than with $\preceq_S$, and the closure and interior operators refer to $\sqsubseteq$.

In general, we consider two players A and B using a channel system $S = (Q, \mathsf{C}, \mathsf{M}, \Delta)$ as their arena. Formally, the set of locations is partitioned into two sets $Q = Q_A \cup Q_B$, one for each player (we do not consider concurrent games). We then let $Conf_A \stackrel{\text{def}}{=} Q_A \times \mathsf{M}^{*\mathsf{C}}$ and $Conf_B \stackrel{\text{def}}{=} Q_B \times \mathsf{M}^{*\mathsf{C}}$ be the recognizable regions where it is A's turn (resp., B's turn) to play.

Goals for the players combine reachability and invariance properties, denoted as above with $\Diamond$ and, respectively, $\Box$. For a goal G, following ATL-like notations [11], we write $\langle A\rangle G$ and $\langle B\rangle G$ for the sets of configurations where, respectively, A and B have a winning strategy to ensure goal G.

In this section, we restrict our attention to *turn-based* games with strict alternation (otherwise we lose decidability, see Remark 43). This means that for all rules $q \xrightarrow{op} q'$ in Δ, $q \in Q_A$ iff $q' \in Q_B$. Thus a single step from some $\sigma \in Conf_A$ can only lead to configurations in $Conf_B$, and reciprocally. In this setting, the following equalities—valid in all

strictly-alternating games—are our main tool for transforming unguarded L_μ terms into equivalent guarded ones:

$$\begin{aligned} &Conf_A \cap Pre(U) = Pre(U \cap Conf_B), \\ &Conf_A \cap \widetilde{Pre}(U) = Conf_A \cap \widetilde{Pre}(U \cap Conf_B). \end{aligned} \tag{41}$$

We omit the symmetrical equalities that allow simplifying for $Conf_B \cap Pre(U)$ etc. For simplicity—and w.l.o.g., see [19]—we assume that players can never be in a deadlock situation where no move is possible, i.e., $\widetilde{Pre}(\emptyset) = \emptyset$.

Remark 43 (Alternation is needed) The strictly-alternating setting adopted in this section is more or less necessary for the decidability results we give below. Indeed, games with no assumption on alternation between players would allow to express temporal model-checking problems, e.g., the CTL formula $\exists\Box R$, for which the support set is not computable.

We could handle a slightly weaker form of alternation, namely games where there exists a uniform bound M on the number of consecutive moves from a same player. The situation is different when the message losses are probabilistic and the stochastic games in Sect. 9 are not required to be strictly alternating.

7.1 Asymmetric games on lossy channel systems

We first consider *asymmetric* games where one player, B, controls the message losses *for both sides*, as well as his perfect moves, while the other player, A, can only perform perfect steps. This setting was previously considered in [5]. It can be used, e.g., to model all the situations where the channel system (player A) must reach some objective against an adversarial environment (player B) responsible for message losses.

B-Reachability games, A-invariance games These are games where player B attempts to reach (visit at least once) a region R, no matter how A behaves. It is known that such games are determined [51]. In our setting, the configurations from which B can achieve the goal $\Diamond R$ can be characterized with:

$$\langle B\rangle\Diamond R = W_B \stackrel{\text{def}}{=} \mu X.R \cup \big(Conf_B \cap Pre(C_\uparrow X)\big) \cup \big(Conf_A \cap \widetilde{Pre}(C_\uparrow X)\big). \tag{42}$$

Here W_B is defined with a guarded term, hence is computable. One way to understand Eq. (42) is to consider a configuration $\sigma \in W_B$. If $\sigma \in Conf A \setminus R$: it is A's turn to play but, since $\sigma \in \widetilde{Pre}(C_\uparrow W_B)$, A can only choose among perfect steps reaching $C_\uparrow W_B$. Now, and since he controls message losses, B can make sure the play ends up in W_B.

We may also consider a different asymmetric setting, where B only controls the message losses *for his moves* while A plays perfect steps where no losses can occur. Then $\langle B\rangle\Diamond R$ is characterized with

$$\langle B\rangle\Diamond R = W_B \stackrel{\text{def}}{=} \mu X.R \cup \big(Conf_B \cap Pre(C_\uparrow X)\big) \cup \big(Conf_A \cap \widetilde{Pre}(X)\big). \tag{43}$$

Here we use Eq. (6), i.e., unfoldings, to deal with the second, unguarded, occurrence of X. This gives

$$\begin{aligned} W_B = \mu X.\big(&R \cup \big(Conf_B \cap Pre(C_\uparrow X)\big) \\ &\cup \big(Conf_A \cap \widetilde{Pre}\big[R \cup \big(Conf_B \cap Pre(C_\uparrow X)\big) \cup \big(Conf_A \cap \widetilde{Pre}(X)\big)\big]\big)\big) \end{aligned}$$

which we now simplify using Eq. (41) and $Conf_A \cap Conf_B = \emptyset$,

$$= \mu X.R \cup \big(Conf_B \cap Pre(C_\uparrow X)\big) \cup \big(Conf_A \cap \widetilde{Pre}\big(Conf_B \cap \big(R \cup Pre(C_\uparrow X)\big)\big)\big)$$
$$= \mu X.R \cup \big(Conf_B \cap Pre(C_\uparrow X)\big) \cup \big(Conf_A \cap \widetilde{Pre}\big(R \cup Pre(C_\uparrow X)\big)\big),$$

and we have a guarded term for W_B. As can be seen, unfolding the recursive definition of W_B exposes terms where the alternation between A and B leads to simplification. This technique is used repeatedly in the rest of this section.

Theorem 44 (Decidability for asymmetric LCS games) *For asymmetric LCS games and recognizable regions R, the sets $\langle B\rangle\Diamond R$ and, dually, $\langle A\rangle\Box R$ are (effective) recognizable regions.*

We have just reproved a result originally from [5]. Note that one can choose whether A steps are lossy (under B's control) or not.

7.2 Symmetric games

We now turn to symmetric games on LCS's, where A and B play in turn, choosing the next configuration. Here both players choose a transition rule to fire and messages to lose after their step.

Reachability games We consider the game where player A tries to reach a set of configurations R whatever the adversarial behavior of B. The set of winning configurations for A is naturally characterized by the following term:

$$\langle A\rangle\Diamond R = W_A \stackrel{\text{def}}{=} \mu X.R \cup \big[Conf_A \cap Pre(C_\uparrow X)\big] \cup \big[Conf_B \cap \widetilde{Pre}(K_\downarrow X)\big].$$

The change from $\widetilde{Pre}(C_\uparrow X)$—previously, in Eq. (42)—or $\widetilde{Pre}(X)$—in Eq. (43)—to $\widetilde{Pre}(K_\downarrow X)$ above reflects the change between an adversary who could choose a firable rule but not his message losses to an opponent who also chooses his message losses. When in $\widetilde{Pre}(K_\downarrow X)$, the adversary ($B$ now) is forced to choose a rule and message losses that will end up in $X = W_A$.

Now, to obtain a guarded term for W_A, we use unfolding and Eq. (41), exactly as we did for asymmetric games:

$$W_A = \mu X.R \cup \big[Conf_A \cap Pre(C_\uparrow X)\big] \cup \big[Conf_B \cap \widetilde{Pre}\big(K_\downarrow\big(R \cup Pre(C_\uparrow X)\big)\big)\big]. \tag{44}$$

Repeated reachability and persistence games In a repeated reachability game, A's goal is to visit region R infinitely many times, whatever the choices of B. (In a persistence game, A aims at remaining inside R from some moment on, no matter how B behaves. Dually, this is a repeated reachability game for B.) In the repeated reachability game, the winning configurations for A are given by

$$\langle A\rangle\Box\Diamond R = W_A \stackrel{\text{def}}{=} \nu Y.\langle A\rangle\Diamond\big(H_A(Y)\big), \quad \text{with}$$
$$H_A(Y) \stackrel{\text{def}}{=} R \cap \big[\big(Conf_A \cap Pre(C_\uparrow Y)\big) \cup \big(Conf_B \cap \widetilde{Pre}(K_\downarrow Y)\big)\big], \tag{45}$$

and where we see $\langle A\rangle\Diamond(\ldots)$ as a unary region-preserving operator given by Eq. (44). Note that $\langle A\rangle\Box\Diamond R$ and $\langle A\rangle\Box(\langle A\rangle\Diamond R)$ do not coincide.

The term in Eq. (45) is not guarded. Furthermore there does not seem to be an easy and direct way to derive an equivalent but guarded term for W_A with unfoldings and simplifications based on Eq. (41). This is because the outermost operator $\langle A\rangle\Diamond$ in W_A makes unfolding harder to manage.

In this situation, we prefer providing alternative definitions for W_A. This can be done by modifying $H_A(Y)$, e.g., $W_A = \nu Y.\langle A\rangle\Diamond H'_A(Y) = \nu Y.\langle A\rangle\Diamond H''_A(Y)$ for

$$H'_A(Y) \stackrel{\text{def}}{=} R \cap \big[\big(Conf_A \cap Pre\big(C_\uparrow \widetilde{Pre}(K_\downarrow Y)\big)\big) \cup \big(Conf_B \cap \widetilde{Pre}(K_\downarrow Y)\big)\big],$$

$$H''_A(Y) \stackrel{\text{def}}{=} Conf_B \cap \big[\big(R \cap \widetilde{Pre}(K_\downarrow Y)\big) \cup \big(\widetilde{Pre}(R \cap K_\downarrow Y)\big)\big].$$

These alternative characterizations rely on the strict alternation of player turns and the fact that the objective, here $\Box\Diamond R$, is prefix-independent.

We now have guarded-term definitions and conclude that reachability, invariance, repeated reachability, and persistence symmetric LCS games are decidable.

Theorem 45 (Decidability for symmetric LCS games) *For symmetric LCS games and recognizable R, the four sets $\langle A\rangle\Diamond R$, $\langle A\rangle\Box R$, $\langle A\rangle\Diamond\Box R$, and $\langle A\rangle\Box\Diamond R$, are (effective) recognizable regions.*

Parity games Parity games on lossy channel systems are briefly mentioned in [5], where it is stated that they can be reduced to a game on a finite arena, namely the one constituted of configurations with empty channels. In fact, this reduction only holds when colors are put on control states (i.e., all configurations with a same given location are colored the same) since then both players are always better off if they systematically empty the channels after their moves. However, this reasoning is not valid when colors are not uniform in control states.

Let us show how to handle the general situation: we consider an arbitrary coloring function $c : Conf \to \{1, \ldots, k\}$ for some $k \in \mathbb{N}$, that assigns a color in $\{1, \ldots, k\}$ with each configuration. We let $color_i \subseteq Conf$ denote the set of configurations colored with color i. Classically, player A wins the game if during the play, the highest color which is visited infinitely often is even. The set W_A of configurations where player A has a winning strategy (against any choices by B) can be characterized in the following way:

$$W_A \stackrel{\text{def}}{=} \lambda_k X_k \ldots \nu X_2.\mu X_1.\left(\left[Conf_A \cap Pre\left(C_\uparrow \bigcup_{i=1}^{k}(color_i \cap X_i)\right)\right] \cup \left[Conf_B \cap \widetilde{Pre}\left(K_\downarrow \bigcup_{i=1}^{k}(color_i \cap X_i)\right)\right]\right),$$

where λ_k is either μ or ν depending on the parity of k.

The above characterization of winning regions holds on any arena [12, Sect. 4.3]. Unfortunately the L_μ term defining W_A is not guarded. However, in the particular case of symmetric LCS games with strict alternation, an alternative characterization is possible:

$$W_A = \lambda_k X_k \ldots \nu X_2.\mu X_1.\left(\left[Conf_A \cap Pre\left(C_\uparrow \widetilde{Pre}\left(K_\downarrow \bigcup_{i=1}^{k}(color_i \cap X_i)\right)\right)\right] \cup \left[Conf_B \cap \widetilde{Pre}\left(K_\downarrow Pre\left(C_\uparrow \bigcup_{i=1}^{k}(color_i \cap X_i)\right)\right)\right]\right).$$

Since W_A can be defined with a guarded term, we conclude that symmetric parity games are decidable on LCS's.

Theorem 46 (Decidability for symmetric parity games) *For symmetric parity games on LCS's and recognizable regions* $(color_i)_{1\leq i\leq k}$, *the set of winning configurations for player A is an* (*effective*) *recognizable region.*

8 Probabilistic lossy channel systems

Our general finite convergence theorem has applications to qualitative probabilistic model-checking problems for well-structured transition systems. As in the previous section about (non-probabilistic) games, lossy channel systems are the prominent example in the literature [1, 3, 15, 16, 71].

We consider here a Markov decision process (MDP) semantics for lossy channel systems where the choice between actions is non-deterministic whereas the losses follow a probabilistic distribution. This model is called NPLCS, for Non-deterministic and Probabilistic LCS.

8.1 The MDP semantics of NPLCS's

We assume familiarity with the basic concepts of MDP's and the algorithmic verification of probabilistic systems. A good introductory exposition is available in [17, Chap. 10].

We view message losses as probabilistic events. Formally, and given a channel system $S = (Q, \mathsf{C}, \mathsf{M}, \Delta)$, we assume a fixed *loss rate* $0 < \lambda < 1$ so that any message waiting inside the channels can be lost with probability λ during the current unit of time, which for simplicity is assumed to coincide with a step of S. Each message is lost or not lost independently of the others. Thus, and given two channel contents $w, w' : \mathsf{C} \to \mathsf{M}^*$, the probability $p_{\text{loss}}(w, w')$ that w becomes w' in one step via message losses is given by $p_{\text{loss}}(w, w') = \#(w', w)\lambda^{|w|-|w'|}(1-\lambda)^{|w'|}$ where $\#(w', w)$ is the number of ways w' can be obtained from w by erasing some symbols, or equivalently, the number of different embeddings of w' into w (see [3, Sect. 5] for details). In particular, $p_{\text{loss}}(w, w') \neq 0$ iff $w' \sqsubseteq w$. These probabilistic losses are lifted to configurations: $p_{\text{loss}}(\sigma, \sigma') = p_{\text{loss}}(w, w')$ if $\sigma = (q, w)$ and $\sigma' = (q, w')$ for some $q \in Q$, and $p_{\text{loss}}(\sigma, \sigma') = 0$ otherwise. This can be seen as a family of probability distributions: for each $\sigma \in \mathit{Conf}_S$, we let $\text{loss}(\sigma) \in \mathit{Dist}(\mathit{Conf}_S)$ be the distribution on Conf_S given by $\text{loss}(\sigma)(\sigma') \stackrel{\text{def}}{=} p_{\text{loss}}(\sigma, \sigma')$.

We may now let the MDP associated with S and λ be $M_S \stackrel{\text{def}}{=} (\mathit{Conf}_S, \to_{\text{mdp}})$ where $\to_{\text{mdp}} \subseteq \mathit{Conf}_S \times \mathit{Dist}(\mathit{Conf}_S)$ is given by

$$\sigma \to_{\text{mdp}} \text{loss}(\tau) \stackrel{\text{def}}{\Leftrightarrow} \sigma \to_{\text{perf}} \tau \text{ is a perfect step of } S \quad \text{(see Sect. 6)}.$$

In other words, the role of the scheduler in M_S is to choose which transition rule to fire, before subsequent losses are decided following the probabilistic distribution. Clearly, when the probabilities are abstracted away from M_S, the resulting steps are exactly the lossy steps given by Eq. (37).

8.2 Qualitative verification of NPLCS's

Natural verification questions for this MDP model are the following: given a linear-time property φ and a starting configuration σ

$$\begin{array}{ll} \text{is there a scheduler } \mathcal{U} \text{ such that } \mathbb{P}_{\mathcal{U}}(\sigma \models \varphi) = 1?, \text{ or} & \text{(almost surely)} \\ \text{is there a scheduler } \mathcal{U} \text{ such that } \mathbb{P}_{\mathcal{U}}(\sigma \models \varphi) > 0? & \text{(positive probability)} \end{array}$$

This is called *qualitative* verification because one only compares $\mathbb{P}_{\mathcal{U}}(\sigma \models \varphi)$ with the extremal values 0 and 1. It turns out that, for NPLCS's, these qualitative properties do not depend on the specific value of the loss rate λ.

On NPLCS's, these problems are undecidable for general LTL formulas, see [3, 15]. In the sequel, we use Theorem 12 to derive decidability for specific subclasses of formulas.

Theorem 47 (Decidability of qualitative model-checking problems for NPLCS [15]) *Given a NPLCS S, an initial configuration σ and a recognizable region R, it is decidable whether there exists a scheduler $\mathcal{U}$ such that*:

$$\begin{array}{llll} \text{(a.1)} & \mathbb{P}_{\mathcal{U}}(\sigma \models \Diamond R) = 0, \text{ or} & \text{(b.1)} & \mathbb{P}_{\mathcal{U}}(\sigma \models \Box\Diamond R) = 1, \text{ or} \\ \text{(a.2)} & \mathbb{P}_{\mathcal{U}}(\sigma \models \Diamond R) < 1, \text{ or} & \text{(b.2)} & \mathbb{P}_{\mathcal{U}}(\sigma \models \Box\Diamond R) < 1, \text{ or} \\ \text{(a.3)} & \mathbb{P}_{\mathcal{U}}(\sigma \models \Diamond R) = 1, \text{ or} & \text{(b.3)} & \mathbb{P}_{\mathcal{U}}(\sigma \models \Box\Diamond R) = 0. \end{array}$$

Furthermore, the sets of configurations from which any of (a.1) *to* (b.3) *holds are recognizable regions effectively computable from R and S.*

To prove Theorem 47, we show that the sets of configurations from which any of (a.1) to (b.3) holds can be expressed by a guarded term in L_μ. By Theorem 12, these sets are computable (assuming that R is a region) and in particular one may decide whether they contain a given initial configuration.

To shorten notations, and given a linear-time property φ, we write $\langle\rangle^{=1}\varphi$ (resp., $\langle\rangle^{>0}\varphi$) for the set of configurations where there is a scheduler such that φ holds *almost surely* (resp., with *positive probability*).

Let us start with (a.1). $\langle\rangle^{=1}\Box R$ can be expressed by a guarded term:

$$\langle\rangle^{=1}\Box R = W \stackrel{\text{def}}{=} \nu X.R \cap Pre(K_\downarrow X). \tag{46}$$

Equation (46) is easy to establish and is not specific to NPLCS's. Note that from $\langle\rangle^{=1}\Box R$ there even exists a *memoryless* scheduler which ensures that *all* executions satisfy $\Box R$. With Eq. (46) and Theorem 12 we conclude that if R is a recognizable region, then $\langle\rangle^{=1}\Box R$ is an (effective) recognizable region, which coincides with (a.1).

The remaining properties, (a.2) to (b.3), also lead to guarded L_μ terms, as witnessed by the following equalities:

$$\langle\rangle^{>0}\Box R = \mu X.\big(\langle\rangle^{=1}\Box R\big) \cup \big(Pre(C_\uparrow X) \cap R\big), \tag{47}$$

$$\langle\rangle^{=1}\Diamond R = \nu X.\mu Y.R \cup Pre(C_\uparrow Y \cap K_\downarrow X), \tag{48}$$

$$\langle\rangle^{=1}\Box\Diamond R = \nu X.\mu Y.Pre\big(C_\uparrow(Y \cup R) \cap K_\downarrow X\big), \tag{49}$$

$$\langle\rangle^{>0}\Diamond\Box R = \mu X.\big(\langle\rangle^{=1}\Box\neg R\big) \cup Pre(C_\uparrow X), \tag{50}$$

$$\langle\rangle^{=1}\Diamond\Box R = \langle\rangle^{=1}\Diamond\big(\langle\rangle^{=1}\Box\neg R\big), \tag{51}$$

and also taking $\langle\,\rangle^{=1}\Box(\ldots)$—defined by Eq. (46)—and $\langle\,\rangle^{=1}\Diamond(\ldots)$—defined by Eq. (48)—as additional unary operators. Since all the terms on the left-hand side are defined with guarded terms, their computability is immediate with Theorem 12.

The proof that Eqs. (47)–(51) are correct—see [15]—is not the topic of this article. Here we just want to observe that, in contrast with Eq. (46), their correctness relies in an essential way on a specific property of the Markov decision processes induced by LCS's, namely, the *finite attractor property*: given a scheduler for M_S there exists a finite set of configurations that is almost-surely visited infinitely often. Moreover, this finite set can be chosen independently from the scheduler:

Lemma 48 (Finite attractor property [13, 76]) *Let $E = \{(q,\varepsilon) \mid q \in Q\}$ be the finite set of configurations where the channels of S are empty. Then $\mathbb{P}_{\mathcal{U}}(\sigma \models \Box\Diamond E) = 1$ for every scheduler $\mathcal{U}$ and starting configuration $\sigma \in Conf_S$.*

The finite attractor property will be used explicitly in the upcoming section.

9 Two-player stochastic games

The NPLCS framework can be seen as a *stochastic game* where a single player, the scheduler, is playing against the probabilistic environment, i.e., against the probabilistic message losses.

It is of course possible to consider 2-player stochastic games on arenas generated by probabilistic LCS's: this question was studied first in [14] and later in [2, 19]. We proceed as in Sect. 7: the two players are called A and B and $Conf_S$ is partitioned as $Conf_A \cup Conf_B$. However, we do not assume anymore that the game is strictly alternating, and therefore $Conf_A$ and $Conf_B$ can be arbitrary regions, but the game is stochastic: once a player has chosen a firable rule, the perfect step is followed by stochastic message losses as in Sect. 8.1.

9.1 Reachability objectives

Let us first consider reachability, or dually invariance, objectives.

Assume A tries to reach region R (goal $\Diamond R$) *with probability 1* no matter how B behaves. The set $\langle A\rangle^{=1}\Diamond R$ of states in which A has an almost-sure winning strategy is given by a guarded term:

$$\langle A\rangle^{=1}\Diamond R = W_A \stackrel{\text{def}}{=} \nu Y.\mu X.\big(R \cup \big[Conf_A \cap Pre(C_\uparrow X \cap K_\downarrow Y)\big] \cup \big[Conf_B \cap \widetilde{Pre}(C_\uparrow X \cap K_\downarrow Y)\big]\big). \tag{52}$$

Justifying Eq. (52) is outside the scope of this article, but we can try to give an intuition of why it works: the inner fixpoint "$\mu X.R \cup \cdots$" define the largest set from which A has a strategy to reach R no matter what B does *if the message losses are favorable*. This may fail if the losses are contrary. However, whatever messages are lost, A's strategy also guarantees that the system will remain in W_A, from which it will be possible to retry the strategy for $\Diamond R$ as many times as necessary. This will eventually succeed almost surely thanks to the finite-attractor property.

Expressions like W_A in Eq. (52) are easier to understand once we introduce a variant of the *Pre* operator that better captures the game-theoretical aspects at hand. Define

$$Pre^{\otimes}_A(X) \stackrel{\text{def}}{=} \big(Conf_A \cap Pre(X)\big) \cup \big(Conf_B \cap \widetilde{Pre}(X)\big), \tag{53}$$

$$Pre^{\otimes}_B(X) \stackrel{\text{def}}{=} \big(Conf_A \cap \widetilde{Pre}(X)\big) \cup \big(Conf_B \cap Pre(X)\big), \tag{54}$$

allowing the reformulation of Eq. (52) as

$$\langle A\rangle^{=1}\Diamond R = W_A \stackrel{\text{def}}{=} \nu Y.\mu X.R \cup Pre^{\otimes}_A(C_{\uparrow}X \cap K_{\downarrow}Y). \tag{52$'$}$$

We note that, since $(Conf_A, Conf_B)$ is a partition of $Conf$, $Pre^{\otimes}_A$ and $Pre^{\otimes}_B$ are dual.

Consider now $\langle A\rangle^{>0}\Diamond R$, where A wants to achieve a reachability objective *with positive probability*. Dually B wants to achieve an invariance objective almost-surely, since these games are determined. Writing R' for $\neg R$, the winning set for B is given by:

$$\langle B\rangle^{=1}\Box R' = W_B \stackrel{\text{def}}{=} \nu X.R' \cap Pre^{\otimes}_B(K_{\downarrow}X). \tag{55}$$

In Eq. (55), the $Conf_B \cap Pre(K_{\downarrow}X)$ component of the $Pre^{\otimes}_B(K_{\downarrow}X)$ accounts for states in which B can choose a perfect move that will end up in $K_{\downarrow}X$, i.e., that can be followed by any adversarial message losses and still remain in X. The $Conf_A \cap \widetilde{Pre}(K_{\downarrow}X)$ component accounts for states in which A cannot avoid going to X, even with favorable message losses.

Equations (52) and (55) give guarded terms for $\langle A\rangle^{=1}\Diamond R$ and $\langle B\rangle^{=1}\Box R'$, so we conclude:

Theorem 49 (Decidability of simple stochastic games on LCS's) *For stochastic LCS-games and recognizable regions R, the sets $\langle A\rangle^{=1}\Diamond R$, $\langle A\rangle^{>0}\Diamond R$, $\langle A\rangle^{>0}\Box R$ and $\langle A\rangle^{=1}\Box R$ are (effective) recognizable regions.*

9.2 Büchi objectives

To date, the main positive result concerning stochastic games on lossy channel systems is the computability of $\langle A\rangle^{=1}\Box\Diamond R$, and $\langle A\rangle^{=1}\bigwedge_{i=1}^{m}\Box\Diamond R_i$, where player A can satisfy a (generalized) Büchi objective, i.e., visit infinitely often a goal region R, or several goal regions $R_1, \ldots, R_m$, almost-surely.

We start with Büchi games. The set $\langle A\rangle^{=1}\Box\Diamond R$ of states from which A has an almost-sure winning strategy can be characterized by the following term:

$$\langle A\rangle^{=1}\Box\Diamond R = W_A \stackrel{\text{def}}{=} \nu Y.\mu X.\big[Y \cap Pre^{\otimes}_A\big(K_{\downarrow}Y \cap C_{\uparrow}(R \cup X)\big)\big]. \tag{56}$$

(See [19] for a justification.) The characterization of the winning region does not provide us with a guarded term: witness the first bound occurrence of Y. However we can replace it with an equivalent guarded term by purely lattice-theoretical reasoning: applying Eq. (9) on W_A gives:

$$\langle A\rangle^{=1}\Box\Diamond R = W'_A \stackrel{\text{def}}{=} \nu Y.\mu X.Pre^{\otimes}_A\big(K_{\downarrow}Y \cap C_{\uparrow}(R \cup X)\big). \tag{57}$$

Theorem 50 (Decidability of stochastic Büchi games on LCS) *Almost-sure Büchi games on stochastic LCS are pure-memoryless determined. Furthermore, if R, $Conf_A$ and $Conf_B$ are recognizable, then the sets of winning positions are (effective) regions too.*

For generalized Büchi games, a similar characterization is possible [19]:

$$\langle A\rangle^{=1}\bigwedge_{i=1}^{m}\Box\Diamond R = W_A \stackrel{\text{def}}{=} \nu Y.\bigcap_{i=1}^{m}\mu X.\big[Y \cap Pre^{\otimes}_A\big(K_{\downarrow}Y \cap C_{\uparrow}(R_i \cup X)\big)\big]. \tag{58}$$

As in Eq. (56), the W_A term is not guarded. Unfortunately, we do not have at hand a generalization of Eq. (9) that would get rid of the m unguarded occurrences of Y. The solution adopted in [19] is to validate a slightly more complex characterization, this times with a guarded term:

$$W_A = W'_A \stackrel{\text{def}}{=} \nu Y.Pre_A^{\otimes}\left(K_{\downarrow}\bigcap_{i=1}^{m}\mu X.\left[Y \cap Pre_A^{\otimes}\left(K_{\downarrow}Y \cap C_{\uparrow}(R_i \cup X)\right)\right]\right). \tag{59}$$

Theorem 51 (Decidability of stochastic generalized Büchi games on LCS) *Almost-sure generalized Büchi games on stochastic LCS are determined. Furthermore, if* $R_1, \ldots, R_m$, $Conf_A$ *and* $Conf_B$ *are recognizable, then the sets of winning positions are (effective) regions too.*

10 Conclusion

We developed a generic criterion, called "upward-guardedness", that guarantees the finite-time convergence of fixpoint equations defining subsets of a well-quasi-ordered set. Our motivations originate in the verification of well-structured transition systems, where the well-known backward-reachability algorithm—see Eq. (2)—is the paradigmatic fixpoint computation that is guaranteed to converge in finite-time. Researchers who extended this technique to more complex and nested fixpoint expressions, where the approximants are not upward- or downward-closed, had to struggle to prove convergence. We hope that the rich sequence of examples given in Sects. 5–9 have convincingly demonstrated the usefulness of our technique for such situations.

So far we did not touch on complexity issues. It is widely believed in the verification community that termination arguments based on WQO theory are not constructive and do not come with complexity bounds. However, the truth is that it is possible to bound the number of steps after which an increasing sequence $V_0 \subseteq V_1 \subseteq V_2 \subseteq \cdots$ of upward-closed subsets must stabilize. This only needs information on the structure of the underlying WQO and a bound on the rate of growth of the sequence of V_i's, often deduced from the complexity of the process that generates it. We refer to [43, 74] for more details: the results there directly apply to our Theorem 12. The complexity upper-bounds one obtains with such WQO-based analysis are often very high, far above the PSPACE, EXPTIME, and nonelementary upper bounds often met in algorithmic verification. Furthermore, these enormous upper bounds are optimal for many WSTS models: this is the case, e.g., for reset/transfer nets, lossy counter machines and lossy channel systems, timed-arc Petri nets and data nets [33, 54, 78] (but not for, e.g., integral relational automata and vector addition systems [29, 41]).

However, these worst-case complexity results are not the real issue in practical applications, for which we believe our results have some significance. Symbolic algorithms directly based on the Finite-Time Convergence Property have been implemented in model checking tools, and the main problem one faces there is not the possibility of terrible time-to-convergence values. Instead, *state explosion* is caused by the need to store and handle "large" V_i sets. Here, dramatic improvements have been obtained by designing improved data-structures, or "symbolic representations", that make better use of symmetries in the data, that are better at sharing common substructures, at caching and recalling instead of recomputing, and that allow more efficient implementations of the required basic operations on upward-closed sets: unions, comparisons, *Pre* and/or *Post*. See, e.g., [36] for $(\mathbb{N}^k, \leq)$, [67] for Presburger-definable regions, and [7] for $(\Sigma^*, \sqsubseteq)$ and channel contents.

Acknowledgements This article is a long and improved version of a conference paper originally coauthored with Christel Baier [14]. It owes a lot to her ideas and contributions but unfortunately she could not join us and sign here because she is co-editing this special issue.

References

1. Abdulla PA, Baier C, Purushothaman Iyer S, Jonsson B (2005) Simulating perfect channels with probabilistic lossy channels. Inf Comput 197(1–2):22–40. doi:10.1016/j.ic.2004.12.001
2. Abdulla PA, Ben Henda N, de Alfaro L, Mayr R, Sandberg S (2008) Stochastic games with lossy channels. In: Proc 11th int conf foundations of software science and computational structures (FOSSACS 2008). Lecture notes in computer science, vol 4962. Springer, Berlin, pp 35–49. doi:10.1007/978-3-540-78499-9_4
3. Abdulla PA, Bertrand N, Rabinovich A, Schnoebelen P (2005) Verification of probabilistic systems with faulty communication. Inf Comput 202(2):141–165. doi:10.1016/j.ic.2005.05.008
4. Abdulla PA, Bouajjani A, Jonsson B, Nilsson M (1999) Handling global conditions in parameterized system verification. In: Proc 11th int conf computer aided verification (CAV 1999). Lecture notes in computer science, vol 1633. Springer, Berlin, pp 134–145. doi:10.1007/3-540-48683-6_14
5. Abdulla PA, Bouajjani A, d'Orso J (2003) Deciding monotonic games. In: Proc 17th int workshop computer science logic (CSL 2003) and 8th Kurt Gödel coll (KGL 2003). Lecture notes in computer science, vol 2803. Springer, Berlin, pp 1–14. doi:10.1007/978-3-540-45220-1_1
6. Abdulla PA, Čerāns K, Jonsson B, Tsay YK (2000) Algorithmic analysis of programs with well quasi-ordered domains. Inf Comput 160(1/2):109–127. doi:10.1006/inco.1999.2843
7. Abdulla PA, Collomb-Annichini A, Bouajjani A, Jonsson B (2004) Using forward reachability analysis for verification of lossy channel systems. Form Methods Syst Des 25(1):39–65. doi:10.1023/B:FORM.0000033962.51898.1a
8. Abdulla PA, Jonsson B (1996) Undecidable verification problems for programs with unreliable channels. Inf Comput 130(1):71–90. doi:10.1006/inco.1996.0083
9. Abdulla PA, Jonsson B (1996) Verifying programs with unreliable channels. Inf Comput 127(2):91–101. doi:10.1006/inco.1996.0053
10. Abdulla PA, Jonsson B, Nilsson M, Saksena M (2004) A survey of regular model checking. In: Proc 15th int conf concurrency theory (CONCUR 2004). Lecture notes in computer science, vol 3170. Springer, Berlin, pp 35–48. doi:10.1007/978-3-540-28644-8_3
11. Alur R, Henzinger T, Kupferman O (2002) Alternating-time temporal logic. J ACM 49(5):672–713
12. Arnold A, Niwiński D (2001) Rudiments of μ-calculus, studies in logic and the foundations of mathematics, vol 146. Elsevier, Amsterdam
13. Baier C, Bertrand N, Schnoebelen P (2006) A note on the attractor-property of infinite-state Markov chains. Inf Process Lett 97(2):58–63. doi:10.1016/j.ipl.2005.09.011
14. Baier C, Bertrand N, Schnoebelen P (2006) On computing fixpoints in well-structured regular model checking, with applications to lossy channel systems. In: Proc 13th int conf on logic for programming and artificial intelligence, and reasoning (LPAR 2006). Lecture notes in computer science, vol 4246. Springer, Berlin, pp 347–361. doi:10.1007/11916277_24
15. Baier C, Bertrand N, Schnoebelen P (2007) Verifying nondeterministic probabilistic channel systems against ω-regular linear-time properties. ACM Transactions on Computational Logic 9(1):5. doi:10.1145/1297658.1297663
16. Baier C, Engelen B (1999) 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 1999). Lecture notes in computer science, vol 1601. Springer, Berlin, pp 34–52. doi:10.1007/3-540-48778-6_3
17. Baier C, Katoen JP (2008) Principles of model checking. MIT Press, Cambridge
18. Bardin S, Finkel A, Leroux J, Schnoebelen P (2005) Flat acceleration in symbolic model checking. In: Proc 3rd int symp automated technology for verification and analysis (ATVA 2005). Lecture notes in computer science, vol 3707. Springer, Berlin, pp 474–488. doi:10.1007/11562948_35
19. Bertrand N, Schnoebelen P (2012) Revisiting stochastic games on lossy channel systems. Submitted for publication
20. Boigelot B (2003) On iterating linear transformations over recognizable sets of integers. Theor Comput Sci 309(1–3):413–468. doi:10.1016/S0304-3975(03)00314-1
21. Boigelot B, Godefroid P (1999) Symbolic verification of communication protocols with infinite state spaces using QDDs. Form Methods Syst Des 14(3):237–255. doi:10.1023/A:1008719024240

22. Boigelot B, Legay A, Wolper P (2003) Iterating transducers in the large. In: Proc 15th int conf computer aided verification (CAV 2003). Lecture notes in computer science, vol 2725. Springer, Berlin, pp 223–235. doi:10.1007/978-3-540-45069-6_24
23. Boigelot B, Legay A, Wolper P (2004) Omega-regular model checking. In: Proc 10th int conf tools and algorithms for the construction and analysis of systems (TACAS 2004). Lecture notes in computer science, vol 2988. Springer, Berlin, pp 561–575. doi:10.1007/978-3-540-24730-2_41
24. Bouajjani A, Habermehl P, Moro P, Vojnar T (2005) Verifying programs with dynamic 1-selector-linked structures in regular model checking. In: Proc 11th int conf tools and algorithms for the construction and analysis of systems (TACAS 2005). Lecture notes in computer science, vol 3440. Springer, Berlin, pp 13–39. doi:10.1007/978-3-540-31980-1_2
25. Bouajjani A, Habermehl P, Vojnar T (2004) Abstract regular model checking. In: Proc 16th int conf computer aided verification (CAV 2004). Lecture notes in computer science, vol 3114. Springer, Berlin, pp 372–386. doi:10.1007/978-3-540-27813-9_29
26. Bouajjani A, Jonsson B, Nilsson M, Touili T (2000) Regular model checking. In: Proc 12th int conf computer aided verification (CAV 2000). Lecture notes in computer science, vol 1855. Springer, Berlin, pp 403–418. doi:10.1007/10722167_31
27. Bouyer P, Markey N, Ouaknine J, Schnoebelen P, Worrell J (2012) On termination and invariance for faulty channel machines. Form Asp Comput 24(4–6):595–607. doi:10.1007/s00165-012-0234-7
28. Bouyer P, Markey N, Reynier PA (2008) Robust analysis of timed automata via channel machines. In: Proc 11th int conf foundations of software science and computational structures (FOSSACS 2008). Lecture notes in computer science, vol 4962. Springer, Berlin, pp 157–171. doi:10.1007/978-3-540-78499-9_12
29. Bozzelli L, Pinchinat S (2012) Verification of gap-order constraint abstractions of counter systems. In: Proc 13th int conf verification, model checking, and abstract interpretation (VMCAI 2012). Lecture notes in computer science, vol 7148. Springer, Berlin, pp 88–103. doi:10.1007/978-3-642-27940-9_7
30. Bradfield JC, Stirling C (2001) Modal logics and mu-calculi: an introduction. In: Bergstra JA, Ponse A, Smolka SA (eds) Handbook of process algebra. Elsevier, Amsterdam, pp 293–330. Chap 4
31. Cécé G, Finkel A, Purushothaman Iyer S (1996) Unreliable channels are easier to verify than perfect channels. Inf Comput 124(1):20–31. doi:10.1006/inco.1996.0003
32. Čerāns K (1994) Deciding properties of integral relational automata. In: Proc 21st int coll automata, languages, and programming (ICALP 1994). Lecture notes in computer science, vol 820. Springer, Berlin, pp 35–46. doi:10.1007/3-540-58201-0_56
33. Chambart P, Schnoebelen P (2008) The ordinal recursive complexity of lossy channel systems. In: Proc 23rd IEEE symp logic in computer science (LICS 2008). IEEE Comput Soc, Los Alamitos, pp 205–216. doi:10.1109/LICS.2008.47
34. Cousot P (2003) Verification by abstract interpretation. In: Verification: theory & practice. Lecture notes in computer science, vol 2772. Springer, Berlin, pp 243–268. doi:10.1007/978-3-540-39910-0_11
35. de Alfaro L, Henzinger TA, Majumdar R (2001) Symbolic algorithms for infinite-state games. In: Proc 12th int conf concurrency theory (CONCUR 2001). Lecture notes in computer science, vol 2154. Springer, Berlin, pp 536–550. doi:10.1007/3-540-44685-0_36
36. Delzanno G, Raskin JF, Van Begin L (2004) Covering sharing trees: a compact data structure for parameterized verification. Int J Softw Tools Technol Transf 5(2–3):268–297. doi:10.1007/s10009-003-0110-0
37. Demri S, Lazić R (2009) LTL with the freeze quantifier and register automata. ACM Trans Computational Logic 10(3):16. doi:10.1145/1507244.1507246
38. Dufourd C, Finkel A, Schnoebelen P (1998) Reset nets between decidability and undecidability. In: Proc 25th int coll automata, languages, and programming (ICALP 1998). Lecture notes in computer science, vol 1443. Springer, Berlin, pp 103–115. doi:10.1007/BFb0055044
39. Emerson EA (1990) Temporal and modal logic. In: van Leeuwen J (ed) Handbook of theoretical computer science, vol B. Elsevier, Berlin, pp 995–1072. Chap 16
40. Emerson EA, Namjoshi KS (1998) On model checking for non-deterministic infinite-state systems. In: Proc 13th IEEE symp logic in computer science (LICS 1998). IEEE Comput Soc, Los Alamitos, pp 70–80. doi:10.1109/LICS.1998.705644
41. Esparza J (1998) Decidability and complexity of petri net problems—an introduction. In: Lectures on Petri nets I: basic models. Lecture notes in computer science, vol 1491. Springer, Berlin, pp 374–428. doi:10.1007/3-540-65306-6_20
42. Fenner S, Gasarch W, Postow B (2009) The complexity of finding SUBSEQ(A). Theory Comput Syst 45(3):577–612. doi:10.1007/s00224-008-9111-4
43. Figueira D, Figueira S, Schmitz S, Schnoebelen P (2011) Ackermannian and primitive-recursive bounds with Dickson's lemma. In: Proc 26th IEEE symp logic in computer science (LICS 2011). IEEE Comput Soc, Los Alamitos, pp 269–278. doi:10.1109/LICS.2011.39

44. Figueira D, Segoufin L (2009) Future-looking logics on data words and trees. In: Proc 34th int symp math found comp sci (MFCS 2009). Lecture notes in computer science, vol 5734. Springer, Berlin, pp 331–343. doi:10.1007/978-3-642-03816-7_29
45. Finkel A (1994) Decidability of the termination problem for completely specified protocols. Distrib Comput 7(3):129–135. doi:10.1007/BF02277857
46. Finkel A, Goubault-Larrecq J (2009) Forward analysis for WSTS, part I: completions. In: Proc 26th ann symp theoretical aspects of computer science (STACS 2009). Leibniz international proceedings in informatics, vol 3. Leibniz-Zentrum für Informatik, Dagstuhl, pp 433–444. doi:10.4230/LIPIcs.STACS.2009.1844
47. Finkel A, McKenzie P, Picaronny C (2004) A well-structured framework for analysing Petri nets extensions. Inf Comput 195(1–2):1–29. doi:10.1016/j.ic.2004.01.005
48. Finkel A, Schnoebelen P (2001) Well-structured transition systems everywhere! Theor Comput Sci 256(1–2):63–92. doi:10.1016/S0304-3975(00)00102-X
49. Glabbeek RJv (1993) The linear time—branching time spectrum II: the semantics of sequential systems with silent moves. In: Proc 4th int conf concurrency theory (CONCUR 1993). Lecture notes in computer science, vol 715. Springer, Berlin, pp 66–81. doi:10.1007/3-540-57208-2_6
50. Glabbeek RJv (2001) The linear time—branching time spectrum I. In: Bergstra JA, Ponse A, Smolka SA (eds) Handbook of process algebra. Elsevier, Amsterdam, pp 3–99. Chap 1
51. Grädel E, Thomas W, Wilke T (eds) (2002) Automata, logics, and infinite games: a guide to current research. Lecture notes in computer science, vol 2500. Springer, Berlin
52. Gruber H, Holzer M, Kutrib M (2009) More on the size of Higman-Haines sets: effective constructions. Fundam Inform 91(1):105–121. doi:10.3233/FI-2009-0035
53. Habermehl P, Vojnar T (2005) Regular model checking using inference of regular languages. In: Proc 6th int workshop on verification of infinite state systems (INFINITY 2004). Electronic notes in theor comp sci, vol 138. Elsevier, Amsterdam, pp 21–36. doi:10.1016/j.entcs.2005.01.044
54. Haddad S, Schmitz S, Schnoebelen P (2012) The ordinal-recursive complexity of timed-arc Petri nets, data nets, and other enriched nets. In: Proc 27th IEEE symp logic in computer science (LICS 2012). IEEE Comput Soc, Los Alamitos, pp 355–364. doi:10.1109/LICS.2012.46
55. Henzinger TA, Majumdar R, Raskin JF (2005) A classification of symbolic transition systems. ACM Trans Comput Log 6(1):1–32. doi:10.1145/1042038.1042039
56. Jančar P (1995) Undecidability of bisimilarity for petri nets and some related problems. Theor Comput Sci 148(2):281–301. doi:10.1016/0304-3975(95)00037-W
57. Jančar P, Esparza J, Moller F (1999) Petri nets and regular processes. J Comput Syst Sci 59(3):476–503. doi:10.1006/jcss.1999.1643
58. Jančar P, Kučera A, Mayr R (2001) Deciding bisimulation-like equivalences with finite-state processes. Theor Comput Sci 258(1–2):409–433. doi:10.1016/S0304-3975(00)00027-X
59. Jeż A, Okhotin A (2012) Representing hyper-arithmetical sets by equations over sets of integers. Theory Comput Syst 51(2):196–228. doi:10.1007/s00224-011-9352-5
60. de Jongh DHJ, Parikh R (1977) Well-partial orderings and hierarchies. Indag Math 39(3):195–207. doi:10.1016/1385-7258(77)90067-1
61. Kesten Y, Maler O, Marcus M, Pnueli A, Shahar E (2001) Symbolic model checking with rich assertional languages. Theor Comput Sci 256(1–2):93–112. doi:10.1016/S0304-3975(00)00103-1
62. Khoussainov B, Nerode A (1995) Automatic presentations of structures. In: Proc int workshop logical and computational complexity (LCC 1994). Lecture notes in computer science, vol 960. Springer, Berlin, pp 367–392. doi:10.1007/3-540-60178-3_93
63. Kouzmin EV, Shilov NV, Sokolov VA (2004) Model checking mu-calculus in well-structured transition systems. In: Proc 11th int symp temporal representation and reasoning (TIME 2004). IEEE Comput Soc, Los Alamitos, pp 152–155. doi:10.1109/TIME.2004.1314433
64. Kruskal JB (1972) The theory of well-quasi-ordering: a frequently discovered concept. J Comb Theory, Ser A 13(3):297–305. doi:10.1016/0097-3165(72)90063-5
65. Kučera A, Schnoebelen P (2006) A general approach to comparing infinite-state systems with their finite-state specifications. Theor Comput Sci 358(2–3):315–333. doi:10.1016/j.tcs.2006.01.021
66. Kunc M (2007) What do we know about language equations? In: Proc 11th int conf developments in language theory (DLT 2007). Lecture notes in computer science, vol 4588. Springer, Berlin, pp 23–27. doi:10.1007/978-3-540-73208-2_3
67. Leroux J, Point G (2009) TaPAS: the Talence Presburger arithmetic suite. In: Proc 15th int conf tools and algorithms for the construction and analysis of systems (TACAS 2009). Lecture notes in computer science, vol 5505. Springer, Berlin, pp 182–185. doi:10.1007/978-3-642-00768-2_18
68. Mayr R (2003) Undecidable problems in unreliable computations. Theor Comput Sci 297(1–3):337–354. doi:10.1016/S0304-3975(02)00646-1

69. Milner EC (1985) Basic WQO- and BQO-theory. In: Rival I (ed) Graphs and order. The role of graphs in the theory of ordered sets and its applications. Reidel, Dordrecht, pp 487–502
70. Perrin D (1990) Finite automata. In: van Leeuwen J (ed) Handbook of theoretical computer science, vol. B. Elsevier, Amsterdam, pp 1–57. Chap 1
71. Rabinovich A (2006) Quantitative analysis of probabilistic lossy channel systems. Inf Comput 204(5):713–740. doi:10.1016/j.ic.2005.11.004
72. Raskin JF, Samuelides M, Van Begin L (2003) Petri games are monotonic but difficult to decide. Tech. report 2003.21, Centre Fédéré en Vérification. Available at http://www.ulb.ac.be/di/ssd/cfv/TechReps
73. Raskin JF, Samuelides M, Van Begin L (2005) Games for counting abstractions. In: Proc 4th int workshop on automated verification of critical systems (AVoCS 2004). Electronic notes in theor comp sci, vol 128. Elsevier, Amsterdam, pp 69–85. doi:10.1016/j.entcs.2005.04.005
74. Schmitz S, Schnoebelen P (2011) Multiply-recursive upper bounds with Higman's lemma. In: Proc 38th int coll automata, languages, and programming (ICALP 2011). Lecture notes in computer science, vol 6756. Springer, Berlin, pp 441–452. doi:10.1007/978-3-642-22012-8_35
75. Schnoebelen P (2001) Bisimulation and other undecidable equivalences for lossy channel systems. In: Proc 4th int symp theoretical aspects of computer software (TACS 2001). Lecture notes in computer science, vol 2215. Springer, Berlin, pp 385–399. doi:10.1007/3-540-45500-0_19
76. Schnoebelen P (2004) The verification of probabilistic lossy channel systems. In: Validation of stochastic systems—a guide to current research. Lecture notes in computer science, vol 2925. Springer, Berlin, pp 445–465. doi:10.1007/978-3-540-24611-4_13
77. Schnoebelen P (2010) Lossy counter machines decidability cheat sheet. In: Proc 4th int workshop reachability problems (RP 2010). Lecture notes in computer science, vol 6227. Springer, Berlin, pp 51–75. doi:10.1007/978-3-642-15349-5_4
78. Schnoebelen P (2010) Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets. In: Proc 35th int symp mathematical foundations of computer science (MFCS 2010). Lecture notes in computer science, vol 6281. Springer, Berlin, pp 616–628. doi:10.1007/978-3-642-15155-2_54
79. Valk R (1978) Self-modifying nets, a natural extension of Petri nets. In: Proc 5th coll automata, languages, and programming (ICALP 1978). Lecture notes in computer science, vol 62. Springer, Berlin, pp 464–476. doi:10.1007/3-540-08860-1_35
80. van Leeuwen J (1978) Effective constructions in well-partially-ordered free monoids. Discrete Math 21(3):237–252. doi:10.1016/0012-365X(78)90156-5
81. Wolper P, Boigelot B (1998) Verifying systems with infinite but regular state spaces. In: Proc 10th int conf computer aided verification (CAV 1998). Lecture notes in computer science, vol 1427. Springer, Berlin, pp 88–97. doi:10.1007/BFb0028736