Abstract interpretation of programs as Markov decision processes

David Monniaux

1 Introduction and related works

The study of probabilistic programs is of considerable interest for the validation of networking protocols, embedded systems, or simply for compiling optimizations. It is also a difficult matter, due to the undecidability of properties on infinite-state deterministic programs, as well as the difficulties arising from probabilities. In this paper, we provide methods for the analysis of programs represented as infinite-state Markov decision processes.

Markov decision processes are a generalization of Markov chains — that is, probabilistic systems where the probabilities of transitions are entirely determined by the last state encountered. They add nondeterministic transitions to the fully probabilistic transitions of the Markov chain. By "nondeterministic transitions", we mean transitions for which an arbitrary choice is made, without any statistical property, as opposed to probabilistic transitions. This greatly enhances the generality of the model, since processes for which all probabilities are not known (for instance, because of an unknown or even hostile environment) can be represented. This extension also allows easier abstraction of Markov chains, since many states with complex probabilistic transitions can be abstracted into fewer states and nondeterministic transitions.

The analysis of *finite-state* Markov decision processes was originally conducted in the fields of operational research and finance mathematics [28]. More recently, they have been studied from

the angle of probabilistic computing systems [1–3, 9, 10, 18, 30]. Effective resolution techniques include linear programming [28, §7.2.7] [6] and newer data structures such as MTBDDs (multi-terminal binary decision diagrams) [1]. However, the problem of large- or infinite-state systems has not been so well studied. The memory size and efficiency gains of BDDs on nondeterministic systems do not apply to computations on probabilistic systems; for this reason, techniques of abstraction and refinement have been recently proposed [9].

In the case of deterministic or nondeterministic systems without a notion of probability, various analysis techniques have been proposed in the last twenty years. Since the problem is undecidable, those techniques are either partially manual (i.e. require the input of *invariants* or similar), either *approximate* (i.e., the analysis takes a pessimistic point of view when it cannot solve the problem exactly). In this paper, we take the latter approach and build our analysis methods upon the existing framework of *abstract interpretation* [8], a general theory of approximation between semantics. We have strived to present the studied problems in an order-theoretic fashion, while some other studies [11] applied advanced probability concepts. The crux of our paper is merely the demonstration of an abstraction relation between two semantics.

We have earlier proposed two classes of automatic methods to analyze such system: some *forward* [19, 20], some *backward* [23, 24]. In this paper, we focus on the backward approach and extend it to a larger class of properties (including those specified by LTL formulas). We also prove that chaotic iterations strategies [7, §2.9] apply to our case, which allows parallel implementations.

In section 2, we give an introduction to probabilistic transition systems, which we extend in section 3 to nondeterministic and probabilistic systems. In section 4, we give a formal language for the specification of trace properties, including those formulated using Büchi or Rabin automata. In section 5, we explain how to

analyze those properties backward and in section 6.1 how to apply abstract analyses. Appendices give mathematical background and necessary lemmas on measure theory, the Lebesgue integral, and lattice theory.

2 Probabilistic transition systems

The natural extension of transition systems to the probabilistic case is *probabilistic transition systems*, also known as *Markov chains* or *discrete-time Markov processes*.

2.1 Probabilistic transitions

We assume that the set of states is *finite* or *countable* so as to avoid technicalities. The natural extension of the notion of deterministic state is the notion of probability distribution on the set of states. We take the convention that a divergent sum of positive numbers, or a divergent integral of a positive function, is equal to $+\infty$.

Definition 1. Let Ω be a finite or countable set of states. A function $f : \Omega \to [0, 1]$ is called a *probability distribution* if $\sum_{\omega\in\Omega} f(\omega) = 1$. We shall note $D(\Omega)$ the set of probabilistic distributions on Ω.

Now that we have the probabilistic counterpart of the notion of state, we need to have the counterpart of the notion of transition.

Definition 2. Let Ω be a finite or countable set of states. Let us consider a function $T : \Omega \times \Omega \to [0, 1]$ such that for all $\omega_1 \in \Omega$, $\sum_{\omega_2\in\Omega} T(\omega_1; \omega_2) = 1$. (Ω, T) is called a *probabilistic transition system*.

If Ω is finite, the relation can be given by a *probabilistic transition matrix*. Let us assimilate Ω to $\{1, \ldots, N\}$. Then, the transition matrix M is defined by $m_{i,j} = T(i, j)$ if $i \to j$, 0 otherwise.

The intuitive notion of a probabilistic transition is that it maps an *input distribution* to an *output distribution*. It is the probabilistic counterpart of the notion of a successor state.

Definition 3. Let T be a transition probability between Ω_1 and Ω_2. Let us define $\overrightarrow{T} : D(\Omega_1) \to D(\Omega_2)$ as follows: $\overrightarrow{T}(d)(\omega_2) = \sum_{\omega_1 \in \Omega_1} T(\omega_1, \omega_2) d(\omega_1)$.

The intuition, using conditional probabilities, is as follows: to each state ω_2 in the arrival state space, we attach the sum of the probabilities of the transitions $\omega_1 \to \omega_2$ for each departure state ω_1; the probability of $\omega_1 \to \omega_2$ is equal to the product of the probability of starting in ω_1 as a departure state and the probability $T(\omega_1, \omega_2)$ of jumping from ω_1 to ω_2 under the hypothesis of starting in ω_1.

Let us now describe the probabilistic counterpart of the notion of predecessor state. Given a transition probability T between Ω_1 and Ω_2 and a boolean property $\pi : \Omega_2 \to \{0, 1\}$, the expectation of a state $\omega_1 \in \Omega_1$ to reach π in one step is then $\sum_{\omega_2 \in \Omega_2} T(\omega_1, \omega_2) \pi(\omega_2)$. We have thus defined a function $\Omega_1 \to [0, 1]$ mapping each state to the corresponding expectation.

A natural extension of this construction is to consider any function $f \in P(\Omega_2) = \Omega_2 \to [0, 1]$. We call such functions *condition functions*.[⋆]

Definition 4. Let T be a transition probability between Ω_1 and Ω_2. Let us define $\overleftarrow{T} : P(\Omega_2) \to P(\Omega_1)$ as follows: $\overleftarrow{T}(f)(\omega_1) = \sum_{\omega_2 \in \Omega_2} T(\omega_1, \omega_2) f(\omega_2)$.

Those functions are linked by the following *adjunction relation*: if T is a transition probability relative to Ω_1 and Ω_2, noting $\langle f, \mu \rangle =$

[⋆] Please note that while those functions look similar to distributions, they are quite different in their meaning and are different mathematical objects when treated in the general, non discrete, case.

$\sum_\omega f(\omega)\mu(\omega)$, then

$$\forall f \in P(\Omega_2)\ \forall\ \mu \in D(\Omega_1)\ \langle f, \overrightarrow{T}.\mu\rangle = \langle \overleftarrow{T}.f, \mu\rangle. \qquad (1)$$

Lemma 5. *For all transition probability T, $\overleftarrow{T}$ is ω-continuous.*

Proof. Let f_n be an ascending sequence. $(\overleftarrow{T}.f_n)(x) = \int f_n \,\mathrm{d}T_x$. The lemma follows from theorem 31. □

2.2 Probability measures on traces

While it is possible to explain probability distributions and transitions on *states* using rather simple mathematical constructions (assuming a finite or countable state space), it is difficult to do so properly on the non-countable sets of infinite traces. For this reason, we shall use the general theory of measures and Lebesgue integration; while it is impossible for reasons of space to recall the details of this theory in this paper, Appendix B.1 presents the definitions and results that we shall use.

We shall use probability measures on *sets of traces* arising from probabilistic transition systems. Let us start with a simple example — consider sequences of tosses of a fair coin: the coin has probability 0.5 of giving 0 and 0.5 of giving 1. A trace is then an infinite sequence of zeroes and ones. Let us consider the (regular) set $0^n(0|1)^*$ of sequences starting by at least n zeroes. It is obvious that the probability of falling into that set is 2^{-n}. The probability of the singleton containing the sequence of only zeroes is 0; actually, in this case, the probability of any singleton set is 0. We see clearly how it is not sufficient to know the probability of all elements in a set to know the probability of an uncountably infinite set.

On the set $\Omega^{\mathbb{N}}$ of infinite traces $(\omega_n)_{n\in\mathbb{N}}$ of elements of Ω we consider the product σ-algebra [26, 27, §III.3]. If $(E_i, \mathcal{A}_i)_{i\in I}$ are measurable sets, the product σ-algebra is the σ-algebra such that the projections $(\pi_i)_{i\in I}$ are measurable. If we consider the σ-algebra

$\mathcal{A}$ on Ω, then the product σ-algebra on $\Omega^{\mathbb{N}}$ is the σ-algebra generated by the cylinders $(\prod_{i<n} A_i) \times \Omega^{\mathbb{N}}$ where $i \in \mathbb{N}$ and the (A_i) are chosen in $\mathcal{A}$.

When taking expectations (or integrals) of functions from traces to real numbers, we shall restrict ourselves to *measurable* functions with respect to this σ-algebra. This is a technical condition; all "interesting" functions we shall consider in §4 have this property. Generally speaking, this measurability will follow from our considering:

- measurable primitive functions;
- point-wise limits of monotone countable sequences of measurable functions; such limits are measurable, and furthermore the integral of the limit of a monotone sequence of functions f_n is the limit of the integrals of the f_n (Th. 31), a result of which we shall make ample usage.

We use the theorem of Ionescu Tulcea (Appendix B.2) to construct the probability measure μ_ω on the set of traces according to the probability distribution μ on the initial states and the transition probability T.

The probability of a property $P : \Omega^{\mathbb{N}} \to \{0, 1\}$ on the traces is then $\int P \, \mathrm{d}\mu_\omega$.

3 Nondeterministic and probabilistic transition systems

We shall see how to combine the notions of *nondeterministic choice* (sets of possible choices for which we know no probabilistic properties) and *probabilistic choice* (sets of possible choices for which we know probabilistic properties), obtaining *discrete-time Markov decision processes* [28] , which has been studied more particularly in the field of operational research and finance mathematics, as well as machine learning.

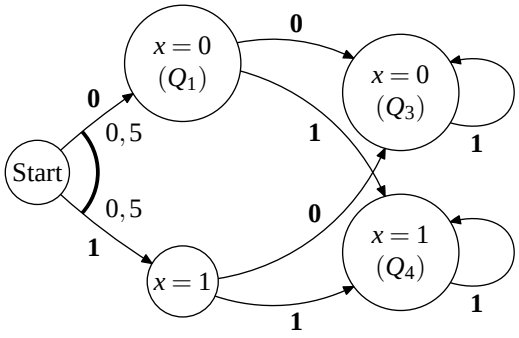
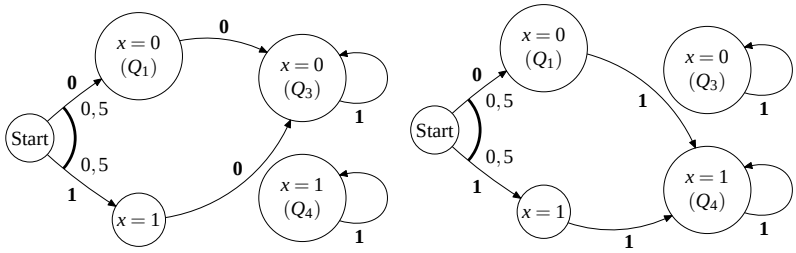


Fig. 1. A probabilistic-nondeterministic transition system



(a) Upper transitions (b) Lower transitions

Fig. 2. Two purely probabilistic transition systems that define, when composed together nondeterministically, the same probabilistic-nondeterministic transition system as in Fig. 1.

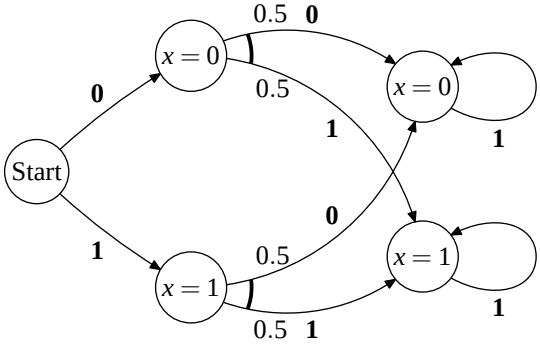


Fig. 3. Another probabilistic-nondeterministic transition system. Note the difference with Fig. 1.

Let us now consider the case where the system must be able to do both nondeterministic and probabilistic transitions (example in Fig. 1). The system then has the choice between different transition probabilities.

For instance, on Fig. 1, in state Q_1, the system has the choice between two partial transition probabilities: the first goes to Q_3 with probability 1, the second goes to Q_4 with probability 1. For an easier intuition, one may think about this choice as if it were

made by an adversary willing to induce certain behaviors. The adversary is supposed to follow a *strategy* or *policy* [28, §2.1.5].

In this paper, we shall assume that the adversary may see the present and past states of the execution, and may act accordingly, thus yielding pessimistic bounds on the outcome of the probabilistic system. In essence, we consider cases where the adversary actively seeks the defeat (or success) of some trace property. Is that a reasonable model? The adversary models the nondeterministic aspects of the external environment of the system, as well as internal factors that are difficult to model precisely, or whose precise implementation is left unknown (e.g. schedulers). Considering that schedulers and a physical environment operate in the worst possible way, rather than according to some statistical properties, generally overestimates the likeliness of problems. However, it seems sensible that an external human user should be modeled in the worst possible case; for instance, the user may actively attack the system in order to obtain certain outcomes.

Let us note that other choices for the power of the adversary can give very different results. For instance, let us consider a program that chooses a Boolean variable x nondeterministically, then chooses a Boolean variable y with uniform probability, then replaces x with the exclusive or (XOR) of x and y (Fig. 3). Clearly, if the nondeterministic chooser is allowed to "look at the future" and predict the outcome of the random generator, it can always arrange for z to be true. If it can only look at the past and the present, or only the present, it cannot and z is uniformly distributed.

Let us suppose that our system is defined by a transition probability T from $\Omega \times Y$ to Ω, where Y is the *domain of nondeterministic choices*. For instance, for the system given in Fig. 1, Y is $\{0, 1\}$ (choice between the upper and lower arrows, as seen in Fig. 2). The operation of the adversary for the transition between states n and $n + 1$ is modeled using an unknown transition probability U_n from Ω^n to Y. The whole transition that is executed by the

system is then the composition $T_n = T \circ \left[\begin{smallmatrix} Id \\ U_n \end{smallmatrix}\right]$, which is a transition probability between Ω^n and Ω. By this notation, we mean that, using the notation of Def. 2,

$$T_n(x_0, \ldots, x_{n-1}; x_n) = T(x_{n-1}, U_n(x_0, \ldots, x_{n-1}); x_n). \quad (2)$$

Ionescu Tulcea's theorem (Appendix B.2) then constructs from the (T_n) a transition probability $G(T, (U_n)_{n\in\mathbb{N}})$ from Ω (the initial state space) to $\Omega^{\mathbb{N}}$.

Let us take a measurable function $f : \Omega^{\mathbb{N}} \to [0, 1]$. $t_0 :: \vec{t}$ will note the infinite trace starting in the state t_0 and following with the infinite sequence $\vec{t}$. To this function we attach its expectation $S_T(f, (U_n)_{n\in\mathbb{N}})$ under the adversary $(U_n)_{n\in\mathbb{N}}$. It is defined as follows:

$$S_T(f, (U_n)_{n\in\mathbb{N}}) = \lambda t_0. \left\langle \lambda \vec{t}. f(t_0 :: \vec{t}), G(T, (U_n)_{n\in\mathbb{N}})(t_0) \right\rangle \quad (3)$$

(S short for S_T if there is no ambiguity about T) and $R(f)$ the set of all functions $S(f, (U_n)_{n\in\mathbb{N}})$ when $(U_n)_{n\in\mathbb{N}}$ is a sequence of transition probabilities, U_n being a transition probability from Ω^n to Y. λ is here a functional notation.

Let E_+^T or, if there is no confusion possible, $E_+(f) = \sup R(f)$ be the *worst-case semantics* and $E_-(f) = \inf R(f)$ be the *best-case semantics* (those designations assume that f indicates some kind of failure condition). Intuitively, if f is the characteristic function of a set of "faulty" traces, E_+ expresses a "worst case" analysis, modeling an adversary willing to make the system err and E_- a "best case" analysis, modeling an adversary willing to prevent the system from erring. $E_+(f)$ is often called the *value* of the Markov decision process with respect to the reward function f (even though we use a slightly different framework as the one given in [28]).

For instance, if we call P the property "the trace ends up with choices of x and y such that x XOR $y = 1$", then for all states σ in Fig. 1, then $E_+(P)(\sigma) = 1$ (it's always possible for the adversary to force going to a state such that x XOR $y = 1$), except, of

course, state Q_3 (where both choices of x and y have been made and x XOR $y = 0$).

Lemma 6. *$E_+(1) = 1$ and $E_+(0) = 0$. E_+ is monotone and upper-continuous.*

4 The properties to analyze

We consider a property to analyze on the traces. To each initial state we attach its *expectation*, that is, the integral of the property to analyze over the traces starting from this state (or the set of integrals, if considering nondeterminism). The properties to analyze are expressed as measurable functions from the set of (possible) traces of execution; we call these functions *trace valuators*. We shall actually consider a class of trace valuators defined syntactically by certain formulas.

4.1 Expectation Functions

Let $I = [0, 1]$ or $[0, +\infty]$, depending on the kind of properties to analyze. Let Ω be a finite or countable set of states — we impose this cardinality constraint so as to avoid theoretical complexities. $\mathcal{P}(\Omega)$ is the set of subsets of Ω. Let $\Omega \to I$ be the set of functions from Ω to I, the *expectation functions* of our system; this set, ordered by $\leq$ point-wise, is a complete lattice. **

4.2 Trace Valuators

Let $\Omega^{\mathbb{N}} \to I$ be the set of measurable functions from $\Omega^{\mathbb{N}}$ to I, ordered point-wise. We shall call such functions "valuators".

**Some of the complexities introduced by uncountably infinite state spaces Ω is that the set of measurable functions from Ω to I is not a complete lattice, but only an ω-complete lattice.

4.2.1 Boolean Trace Valuators

We take $I = [0, 1]$ or even $I = \{0, 1\}$.

We shall consider formulas written in the following language:

$$\begin{aligned} \mathit{formula} ::= {} & \mathit{name} \\ & \mid \mathit{constant} \\ & \mid \mathit{formula}_1 +_{\mathit{set}} \mathit{formula}_2 \text{ where } \mathit{set} \subseteq \Omega \\ & \mid \mathit{constant} +_{\mathit{set}} \mathit{formula} \\ & \mid \mathrm{lfp}(\mathit{name} \mapsto \mathit{formula}) \\ & \mid \mathrm{gfp}(\mathit{name} \mapsto \mathit{formula}) \\ & \mid \mathrm{shift}(\mathit{formula}) \\ & \mid \text{let } \mathit{name} = \mathit{formula}_1 \text{ in } \mathit{formula}_2 \end{aligned}$$

Let $\mathrm{shift} : \Omega^{\mathbb{N}} \to \Omega^{\mathbb{N}}$: $(\mathrm{shift}.\vec{t})_k = t_{k+1}$ be the function that chops off the first element of an infinite trace.

Let env_t be the set of environments of valuators, mapping each *name* to a valuator, ordered point-wise.

$[\![\mathit{formula}]\!]_t : \mathit{env}_t \to (\Omega^{\mathbb{N}} \to I)$ is defined inductively as follows:

$$[\![\mathit{name}]\!]_t.\mathit{env} = \mathit{env}(\mathit{name}) \tag{4}$$

$$[\![\mathit{constant}]\!]_t.\mathit{env} = \lambda\vec{t}.\mathit{constant} \tag{5}$$

$$[\![f_1 +_S f_2]\!]_t.\mathit{env} = \lambda\vec{t}.\,\chi_S(t_0).([\![f_1]\!]_t.\mathit{env}(\vec{t})) + \chi_{S^C}(t_0).([\![f_2]\!]_t.\mathit{env}(\vec{t})) \tag{6}$$

$$[\![\mathrm{lfp}(\mathit{name} \mapsto f)]\!]_t.\mathit{env} = \mathrm{lfp}(\lambda\phi.[\![f]\!]_t.\mathit{env}[\mathit{name} \mapsto \phi]) \tag{7}$$

$$[\![\mathrm{gfp}(\mathit{name} \mapsto f)]\!]_t.\mathit{env} = \mathrm{gfp}(\lambda\phi.[\![f]\!]_t.\mathit{env}[\mathit{name} \mapsto \phi]) \tag{8}$$

$$[\![\mathrm{shift}(f)]\!]_t.\mathit{env} = ([\![f]\!]_t.\mathit{env}) \circ \mathrm{shift} \tag{9}$$

$$[\![\text{let } \mathit{name} = f_1 \text{ in } f_2]\!]_t.\mathit{env} = [\![f_2]\!]_t.\mathit{env}[\mathit{name} \mapsto [\![f_1]\!]_t.\mathit{env}] \tag{10}$$

χ_S is the characteristic function of S and S^C the complement of S. $t_0 \in \Omega$ is the first state of the sequence of states $\vec{t} \in \Omega^{\mathbb{N}}$.

Lemma 7. *For all formula f, $[\![f]\!]_t$ is monotone. For all formula*

f without lfp *or* gfp, *$[\![f]\!]_t$ is continuous. (Def. 16)*

Proof. Obvious by induction on the structure of f. □

4.2.2 Some particularly interesting Boolean valuators

We shall consider in this section four very important classes of properties, all of which can be shown to be measurable.

- Let A be a set of states. The *reachability* property associated with A defines the set of traces that pass through A at some point. It corresponds to the formula:

$$\mathrm{lfp}(f \mapsto 1 +_A \mathrm{shift} f) \tag{11}$$

  $[\![\mathrm{lfp}(f \mapsto 1 +_A \mathrm{shift} f)]\!]_t$ is defined as the least fixpoint of Φ, where $\Phi(W)$ is the set of traces t such that $t_0 \in A$, or shift$t \in W$. Φ is clearly continuous, therefore $\mathrm{lfp}\,\Phi = \bigcup_{n=0}^{\infty} \Phi^n(\emptyset)$. $\Phi^n(\emptyset)$ is the set of traces t such that $\exists k < n\ t_k \in A$.
- Let A be a set of states. The *liveness* property associated with A defines the set of traces that always remain in A. It corresponds to the formula:

$$\mathrm{gfp}(f \mapsto \mathrm{shift} f +_A 0) \tag{12}$$

  This is the dual of the preceding definition: $[\![f \mapsto \mathrm{shift} f +_A 0]\!]_t$ is $\bigcap_{n=0}^{\infty} \Psi^n(\Omega^{\mathbb{N}})$, where $\Psi^n(\Omega^{\mathbb{N}})$ is the set of traces t such that $\forall k < n\ t_k \in A$.
- Let A be a (measurable) set of states. The *Büchi acceptance* property associated with A defines the set of traces that pass through A infinitely often; it is written as:

$$\mathrm{gfp}(C \mapsto \mathrm{lfp}(R \mapsto \mathrm{shift}(C) +_A \mathrm{shift}(R))) \tag{13}$$

  The *co-Büchi* property is just the negation of this formula.

  Let us recall that a Büchi automaton [32, chapter 4, part I] on an alphabet A is given by a state space Σ, an initial state σ_0, a set $F \subseteq \Sigma$ of final states and a transition relation $T \subseteq \Sigma \times A \times \Sigma$. An infinite input sequence $t_0, t_1, \ldots$ is accepted by

the automaton if there exists a sequence $\sigma_0, \sigma_1, \ldots$ of states such that for all i, $(\sigma_i, t_0, \sigma_{i+1}) \in T$ and there exists an infinity of i such that $t_i \in F$.

If a trace property is defined using a deterministic Büchi automaton $\mathcal{A}$, then this property may be defined for a Markov decision process S as the Büchi acceptance B of the set $\Omega \times F$ over the synchronous product of S and A, where F is the set of final states of $\mathcal{A}$. By synchronous product, we mean that the Büchi automaton is made to read the state of S as an input sequence.

Note, however, that this does not extend to nondeterministic Büchi automata; that is, in the case of nondeterministic Büchi automata $\mathcal{A}$, $E_+(B)$, as defined above, is *not* the probability in the worst case of generating a trace accepted by $\mathcal{A}$. Because of this discrepancy, and the fact that deterministic Büchi automata are strictly less powerful than nondeterministic ones [32, chapter 4, §I.4], we shall be forced to consider deterministic Rabin automata (see next point).

We shall now briefly see the reasons for this discrepancy. $E_+(B)$ is the upper bound, over the strategies of the adversary and the nondeterministic Büchi automaton $\mathcal{A}$, of the probability of passing through $\Omega \times F$ infinitely often; while the desirable property is the upper bound over the strategies of the adversary of the probability of finding a strategy for the Büchi automaton to accept the trace. If we fix the adversary A, the property is written as:

$$\sup_{S \text{ strategy of the Büchi automaton}} \mathbb{P}\left(\vec{t} \mapsto \mathcal{A} \text{ accepts } \vec{t} \text{ using strategy } S\right), \tag{14}$$

while the latter property is:

$$\mathbb{P}\left(\vec{t} \mapsto \begin{array}{l} \exists S \text{ strategy of the Büchi automaton} \\ \mathcal{A} \text{ accepts } \vec{t} \text{ using strategy } S \end{array}\right) \tag{15}$$

In the latter case, the Büchi automaton can choose its strategy depending on the knowledge of the full trace $\vec{t}$; this would cor-

respond to a model where the adversary "looks at the future" (see §3). However, in the former case, the Büchi automaton is restricted to strategies where only the past history of the computation is known.

- Given a sequence $\emptyset \subseteq U_{2k} \subseteq \cdots \subseteq U_0 = \Omega$, the *Rabin acceptance property* associated with (U_n) is the set of traces defined by the following temporal property [11, sect. 5]:

$$\mathcal{R} = \bigvee_{i=0}^{k-1} (\Box\Diamond U_{2i} \wedge \neg\Box\Diamond U_{2i+1}) \tag{16}$$

It corresponds to the following formula:

$$\begin{aligned} \mathrm{gfp}(x_{2k-1} \mapsto \mathrm{lfp}(x_{2k} \mapsto \cdots \mathrm{gfp}(x_1 \mapsto \mathrm{lfp}(x_0 \mapsto \\ ((\cdots (x_{2k-1} + U_{2k-1} x_{2k-2}) \cdots +_{U_1} x_0) +_{U_0} 0) \end{aligned} \tag{17}$$

Deterministic Rabin automata are equivalent to Muller automata, from McNaughton's theorem [32, Th 4.4], which are equivalent to nondeterministic Büchi automata and ω-regular languages (op. cit.). In §4.3.1, we shall see a way to use them to represent linear time properties.

4.2.3 Summation valuator

A related family of trace valuators are the *summing* valuators. The summation valuator associated with a (measurable) function $f : \Omega \mapsto [0, +\infty]$ is the function

$$[\![\Sigma A]\!]_t : \left| \begin{array}{l} \Omega^{\mathbb{N}} \quad \to [0, +\infty] \\ (x_n)_{n\in\mathbb{N}} \mapsto \sum_{k=0}^{\infty} f(x_k) \end{array} \right. \tag{18}$$

Essentially, a summation valuator assigns to each state a "prize" and sums the prizes along a trace.

Obviously, this function can be formulated as a least fixpoint:

$$[\![\Sigma f]\!]_t = \mathrm{lfp}(\phi \to f + \phi \circ \mathrm{shift}) \tag{19}$$

This construct has two obvious applications:

- counting the average number of times the program goes through a certain set of states A; here, f is the characteristic function of A;
- counting the average time spent in the process; here f maps each state to the amount of time spent in that state (0 for states meaning "termination").

For instance, to evaluate the average number of time that a program goes through a loop whose head label is marked with $l \in L$, where L is the set of labels and M is the set of possible memory environments, one applies the summation valuator associated with the characteristic function of $\{l\} \times M$.

4.3 Temporal logics

Temporal logics [4, chapter 3] are expressive means of specifying properties of transition systems.

4.3.1 Linear time logic (LTL) and ω-regular conditions

A formula F in LTL defines an ω-regular set of traces $[\![F]\!]_t$, that is, a set of traces recognizable by a (nondeterministic) Büchi automaton B [4, §3.2], or, equivalently, by a deterministic Rabin automaton R [31, §4].

Let us consider a (nondeterministic) probabilistic transition system T, and the according definition of E_+^T. Let us consider the synchronous product $T \times R$, and C the associated Rabin acceptance condition. $E_+^T([\![F]\!]_t)$ is then equal to $E_+^{S\times R}([\![C]\!]_t)$ [11, §5].

If B is deterministic, we can similarly consider the synchronous product $T \times B$, and C the associated Büchi condition. $E_+^T([\![F]\!]_t)$ is then equal to $E_+^{S\times B}([\![C]\!]_t)$ [11, §4].***

*** Note that this does not hold for nondeterministic Büchi automata, since the

*4.3.2 Branching-time logic: pCTL and pCTL**

The branching-time logics CTL and CTL* [4, §3.2] have had much success in the analysis of nondeterministic (albeit non probabilistic) systems. It was therefore quite natural to extend this notion to probabilistic systems. Proposed extensions to the probabilistic case include pCTL [12] and pCTL*. We shall see here briefly how we deal with some pCTL* formulas.

CTL* formulas define sets of states as the starting states of sets of traces defined by LTL path formulas (in which state formulas are CTL* state formulas).

The operation that makes a CTL* state formula out of a LTL path formula is the taking of the initial states: if $[\![\mathtt{f}]\!]_s$ denotes the semantics of f as a state formula and $[\![\mathtt{f}]\!]_p$ its semantics as a path formula, then

$$[\![\mathtt{f}]\!]_s = \{x_0 \in \Omega \mid \exists x_1, \ldots \ \langle x_0, x_1, \ldots\rangle \in [\![\mathtt{f}]\!]_p\}. \tag{20}$$

In the case of probabilistic systems, we do not have sets of starting states but expectation functions; such expectation functions are then compared to a threshold value, which gives sets of states. State formulas noted as $f_{\bowtie\alpha}$ are thus obtained, where f is a trace valuator and $\bowtie$ is $\leq$, $<$, $=$, $>$ or $\geq$. The semantics of this construct is as follow:

$$[\![f_{\bowtie\alpha}]\!] = \{x_0 \in \Omega \mid \forall (U_n)_{n\in\mathbb{N}} S([\![f]\!]_t\,, (U_n)_{n\in\mathbb{N}})(x_0) \bowtie \alpha\} \tag{21}$$

In the case of $<$ and $\leq$, giving an upper bound on those sets is easy provided we have an upper bound $[\![f]\!]^\sharp_{e+}$ of $E_+([\![t]\!]_t)$ (see §5):

$$\forall x_0 \, [\![f]\!]^\sharp_{e+}(x_0) \bowtie \alpha \implies x_0 \notin [\![f_{\bowtie\alpha}]\!]\,. \tag{22}$$

We think that the intuitive meaning of LTL formulas is clearer

automaton is allowed to take its nondeterministic choices with the knowledge of the full sequence of states, not only the past and present states.

Program:

```
/* A */
if (x < 0.5) /* true */  x += uniform();
else         /* false */ x -= uniform();
/* B */
post_condition (x>=0.9 && x<= 1.1);
/* C */
```

`uniform()` uniform in [0,1].
Property: reachability of C.
Program points:

- A, B, C
- beginning of both branches of the test true and false

Backward computation:

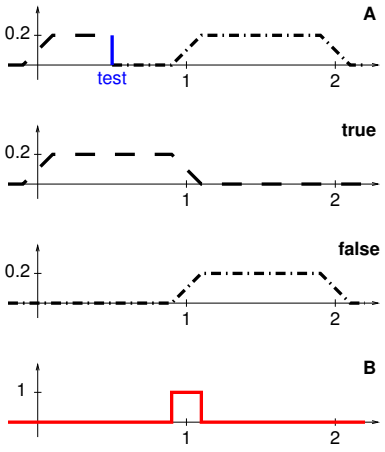


Fig. 4. An example of backward upper semantics: the state space is $\Pi \times \mathbb{R}$ where Π is the set of program points, and the single variable x is chosen in $\mathbb{R}$.

than that of non-trivial pCTL* formulas. However, it is an added benefit of our method that it can analyse programs with respect to certain pCTL* formulas (those for which state formulas may be analysed by obtaining upper bounds of $E_+(P)$ where P is a trace formula).

5 Backwards Worst Case Analysis

In section 4.2, we gave the syntax and semantics of a logic describing sets of traces, or, more generally, measurable functions over the traces. Given a formula f, one may want to compute its worst-case probability $E_+([\![f]\!]_t)$, or at least get an upper bound for it. Unfortunately, the definitions of both $[\![f]\!]_t$ and E_+ do not yield effective means to do so.

In §5.1 we shall attach to each formula f another semantics $[\![f]\!]_{e+}$, which we shall show how to abstract in §6. We shall see the abstraction relationship between $E_+([\![f]\!]_t)$ and $[\![f]\!]_{e+}$ in §5.2.

A well-known solution to the problem of the optimal value of a Markov decision process is *value iteration* [28, §7.2.4]. This method is mainly of theoretical interest for the analysis of finite state Markov decision processes, since there is little control as to its rate of convergence and much better algorithms are available [28, §7.2.4]. On the other hand, since it is actually a kind of generalization to Markov decision processes of the backwards reachability analysis for nondeterministic systems, we can apply abstract interpretation techniques so as to provide an effective mean to compute upper bounds on the probability of the properties to analyze.

We shall now define another semantics $[\![P]\!]_{e+}$, which will assign to each initial state a worst-case probability, or an upper bound thereof, of starting a trace fulfilling the property P. Since this semantics starts from the final states of the computation and then walks back the transitions, we call it a *backwards* semantics. In section 7, we shall discuss this choice of a backward semantics, whereas several already proposed semantics for probabilistic programs were given in a forward fashion. Let us still note that forward and backward probabilistic denotational semantics for programs are equivalent [23].

We shall use the program on Fig. 4 as a simple running example. The most complex part of the semantics of that program in terms of a Markov decision process is the semantics of `x += uniform();` (and similarly for `-=`). This operation may be divided into three steps:

- compute a random number y according to `uniform()` ($\mathrm{d}\mu_{\texttt{uniform}}$ is the probability measure of this generator);
- compute `x = x +` y`;`;
- discard y.

The latter two steps are compounded into: compute the linear

transformation $(x, y) \mapsto x$. The compound transition of this operation T is somewhat complex. However, we shall only be interested in its associated backward operator $\overleftarrow{T}$, which we compute compositionally [23]:

$$\overleftarrow{T}(f) = \begin{pmatrix} (\mathbb{R} \times \mathbb{R} \to \mathbb{R}) \to (\mathbb{R} \to \mathbb{R}) \\ g \qquad\qquad\quad \mapsto (x \mapsto \int g(x, y)\ \mathrm{d}\mu_{\texttt{uniform}}) \end{pmatrix} \circ \begin{pmatrix} (\mathbb{R} \to \mathbb{R}) \to (\mathbb{R} \times \mathbb{R} \to \mathbb{R}) \\ h \qquad\quad \mapsto ((x, y) \mapsto h(x, y)) \end{pmatrix} \tag{23}$$

Let env_e be the set of environments of expectation functions (an environment of expectation functions maps each *name* to a expectation function), ordered point-wise.

$[\![formula]\!]_{e+} : (\Omega \to I) \to (\Omega \to I)$ is defined inductively as follows:

$$[\![name]\!]_{e+}.env = env(name) \tag{24}$$

$$[\![constant]\!]_{e+}.env = \lambda x.constant \tag{25}$$

$$[\![f_1 +_S f_2]\!]_{e+}.env = \chi_S.([\![f_1]\!]_{e+}\ env) + \chi_{S^C}.([\![f_2]\!]_{e+}\ env) \tag{26}$$

$$[\![\mathrm{lfp}(name \mapsto f)]\!]_{e+}.env = \mathrm{lfp}(\lambda\phi.[\![f]\!]_{e+}.env[name \mapsto \phi]) \tag{27}$$

$$[\![\mathrm{gfp}(name \mapsto f)]\!]_{e+}.env = \mathrm{gfp}(\lambda\phi.[\![f]\!]_{e+}.env[name \mapsto \phi]) \tag{28}$$

$$[\![\mathrm{shift}(f)]\!]_{e+}.env = \sup_{T \in \mathcal{T}}(\overleftarrow{T}([\![f]\!]_{e+}.env)) \tag{29}$$

$$[\![\text{let } name = f_1 \text{ in } f_2]\!]_{e+}.env = [\![f_2]\!]_{e+}.env[name \mapsto [\![f_1]\!]_{e+}.env] \tag{30}$$

This semantics is thus some form of μ-calculus, except that "lfp" replaces the μ binder and "gfp" ν; but since we also use μ to note measures, it would have been confusing to also use it in the syntax of the formulas.

Let us see what this computation yields on the program in Fig. 4 and the trace property $\mathrm{lfp}(f \mapsto 1 +_C f)$. The state space is $\Omega = \Pi \times \mathbb{R}$ where $\Pi = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{true}, \mathsf{false}\}$ is the set of program points and $\mathbb{R}$ is the set of all possible memory configurations (which we reduce to a single real variable `x`; for the sake of simplicity, we choose here to use real numbers instead of machine-representable values). On the right of the figure, we have represented various functions f_i for $i \in \Pi$, each representing a part of the final invariant. For the sake of simplicity of notation, we shall note f_i both a function $\mathbb{R} \to [0,1]$ and its extension in $\Pi \times \mathbb{R} \to [0,1]$ defined as follows: $f_i(i,x) = f_i(x)$ and $f_i(j,x) = 0$ for $j \neq i$.

For instance, f_{true} is obtained as the function $x \mapsto \int f_b(x + y)\, \mathrm{d}\mu_{\mathrm{uniform}}(y)$. The iterations of the least fixpoint are as follows: $u_0 = 0$, $u_1 = f_{\mathsf{C}}$, $u_2 = u_1 + f_{\mathsf{B}}$, $u_3 = u_2 + f_{\mathsf{true}} + f_{\mathsf{false}}$, $u_4 = u_3 + f_{\mathsf{A}}$, and the fixpoint is reached.

As for the summation valuator,

$$[\![\Sigma f]\!]_{e+} = \mathrm{lfp}\left(\phi \mapsto f + \sup_{T \in \mathcal{T}} (\overleftarrow{T}.\phi)\right) \tag{31}$$

Lemma 8. *Let f be a formula, $[\![f]\!]_{e+}$ is monotone.*

Lemma 9. *Let f be a formula not containing* gfp. *$[\![f]\!]_{e+}$ is ω-upper-continuous.*

Proof. By structural induction on f. Let env_n be an ascending sequence of environments, whose limit is env_∞. The cases for *name*, *constant* and "let" are trivial.

- Let $t_0 \in S$. $\sqcup_{n \in \mathbb{N}} \underbrace{([\![f_1 +_S f_2]\!]_{e+}.env_n)(t_0)}_{([\![f_1]\!]_{e+}.env)(t_0)} = ([\![f_1]\!]_{e+}.env_\infty)(t_0) = ([\![f_1 +_S f_2]\!]_{e+}.env_\infty)(t_0)$. Similarly for $t_0 \notin S$.

- The shift case:

$$\begin{aligned}
&[\![\mathrm{shift}(f)]\!]_{e+}.env_\infty = \sqcup_{T\in\mathcal{T}} \overleftarrow{T}([\![f]\!]_{e+}.env_\infty)) \\
&= \sqcup_{T\in\mathcal{T}} \overleftarrow{T}(\sqcup_n [\![f]\!]_{e+}.env_n) = \sqcup_{T\in\mathcal{T}} \sqcup_n \overleftarrow{T}.([\![f]\!]_{e+}.env_n) \\
&= \sqcup_n \sqcup_{T\in\mathcal{T}} \overleftarrow{T}.([\![f]\!]_{e+}.env_n) = \sqcup_n [\![\mathrm{shift}(f)]\!]_{e+}.env_n
\end{aligned}$$

- The lfp case follows by applying lemma 22 to the induction hypothesis. □

5.2 The Abstraction Relation Between the Semantics

In §4.2 we described a semantics ($[\![\cdot]\!]_t$) mapping formulas to measurable functions defined on the infinite traces; in §3 we defined E_+ mapping such functions to expectation functions on the initial states, depending on the (nondeterministic) probabilistic transition system to be studied. In §5.1, we directly mapped the formulas to expectation functions using another semantics ($[\![\cdot]\!]_{e+}$). What is the relation between those two ways to associate an expectation function to each formula?

We shall see that $[\![f]\!]_{e+}$ is an abstraction of $[\![f]\!]_t$ with respect to the abstraction map E_+, and that it is actually optimal when f does not contain greatest fixpoints.

Before stating the main results, we shall first go through a few technical lemmas.

Lemma 10. *For all f, t_0 and U_1,*

$$\begin{aligned}
&\sup_{(U_n)_{n\geq 2}} \int \lambda\langle t_1, \ldots\rangle.f(\langle t_1, \ldots\rangle)\,\mathrm{d}[G(T,(U_n)_{n\in\mathbb{N}}).t_0] \\
&= \sup_{(U_n)_{n\geq 2}\ U_n \text{ does not depend on } t_0} \\
&\qquad \int \lambda\langle t_1, \ldots\rangle.f(\langle t_1, \ldots\rangle)\,\mathrm{d}[G(T,(U_n)_{n\in\mathbb{N}}).t_0]
\end{aligned}$$

Proof. Let a and b be respectively the left-hand and right-hand sides of the equality to be proven. $a \geq b$ is obvious since b is an upper bound on a smaller set than a. Let us now prove $b \geq a$. Let us take $t_0 \in X$ and $(U_n)_{n\geq 2}$. Let us now consider $(\tilde{U}_n)_{n\geq 2}$ defined by $\tilde{U}_n(\vec{t'}; W) = U_n.(t_0, t'_1, \ldots, t'_{n-1}; W)$. $\tilde{U}_n(t'_0, \ldots; W)$ does not depend on t'_0. Furthermore,

$$\int \lambda\langle t_1, \ldots\rangle.f(\langle t_1, \ldots\rangle)\,\mathrm{d}[G(T, (U_n)_{n\in\mathbb{N}}).t_0] \\ = \int \lambda\langle t_1, \ldots\rangle.f(\langle t_1, \ldots\rangle)\,\mathrm{d}[G(T, (\tilde{U}_n)_{n\in\mathbb{N}}).t_0].$$

Thus $a(t_0) \leq b(t_0)$. □

Lemma 11. *For all t_0 in$X$,*

$$E_+(\lambda t.\chi_S(t_0).V_1(t) + \chi_{S^C}(t_0).V_2(t)).t_0 \\ = \chi_S(t_0).(E_+(V_1).t_0) + \chi_{S^C}(t_0).(E_+(V_2).t_0). \quad (32)$$

Proof. Let $t_0 \in X$.

Let us suppose that $t_0 \in S$. Then $E_+(\lambda t.\chi_S(t_0).V_1(t) + \chi_{S^C}(t_0).V_2(t)).t_0 = E_+(V_1).t_0 = \chi_S(t_0).(E_+(V_1).t_0) + \chi_{S^C}(t_0).(E_+(V_2).t_0)$. Similar proof for $t_0 \notin S$.

□

Lemma 12. *For any trace valuator f, for any g_1 and g_2 in $R(f)$, for any $A \subseteq \Omega$, the function g_3 defined by $g_3(\vec{t}) = g_1(\vec{t})$ if $t_0 \in A$, $g_3(\vec{t}) = g_2(\vec{t})$ otherwise, belongs to $R(f)$.*

Proof. Let $g_1 = S(f, (U^1_n)_{n\in\mathbb{N}})$ and $g_2 = S(f, (U^2_n)_{n\in\mathbb{N}})$. Let us define $(U^3_n)_{n\in\mathbb{N}}$ as follows: $U_3(\vec{t}, W) = U_1(\vec{t}, W)$ if $t_0 \in A$, $U_2(\vec{t}, W)$ otherwise. Let $g_3 = S(f, (U^3_n)_{n\in\mathbb{N}})$. Obviously, if $t_0 \in A$, then $g_3(t_0) = g_1(t_0)$, otherwise $g_3(t_0) = g_2(t_0)$. □

The next theorem gives a link between $[\![f]\!]_{e+}$ and $[\![f]\!]_{e+}$ when f does not contain greatest fixpoints; namely, $E_+([\![f]\!]_{e+}) = [\![f]\!]_{e+}$ if f has no free variables. In the context of Fig. 4, this means

that each of our functions f_i ($i \in \Pi$) maps each x to the maximal probability that it starts a trace leading to C.

Theorem 13. *Let f be a formula not containing* gfp. *Let env be an environment of valuators. Noting $E_+(env)$ the point-wise application of E_+ to env,*

$$\llbracket f \rrbracket_{e+} .(E_+(env)) = E_+(\llbracket f \rrbracket_t .env) \tag{33}$$

Proof. Proof by induction on the structure of f.

- The cases for "let", *name* and *constant* are trivial.
- For $f_1 +_S f_2$: Let $t_0 \in X$.

$$\begin{aligned}
&\llbracket f_1 +_S f_2 \rrbracket_{e+} .(E_+(env)).t_0 \\
&= \chi_S(t_0).(\llbracket f_1 \rrbracket_{e+} .E_+(env).t_0) + \chi_{S^C}(t_0).(\llbracket f_2 \rrbracket_{e+} .E_+(env).t_0) \\
&= \chi_S(t_0).(E_+(\llbracket f_1 \rrbracket_t .env).t_0) + \chi_{S^C}(t_0).(E_+(\llbracket f_2 \rrbracket_t .env).t_0) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(induction)} \\
&= E_+(\lambda t.\chi_S(t_0).(\llbracket f_1 \rrbracket_t \, env t) + \chi_{S^C}(t_0).(\llbracket f_2 \rrbracket_t \, env t)).t_0 \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(lemma 11)} \\
&= E_+(\llbracket f_1 +_S f_2 \rrbracket_t).t_0.
\end{aligned}$$

- For shift: Let us first fix U_1. Let us note $T_1 = T \circ \left[\begin{smallmatrix}\mathrm{Id} \\ U_1\end{smallmatrix}\right]$ and consider $\overleftarrow{T_1}.E_+\left(\llbracket f \rrbracket_t\right)$. From lemma 5, $\overleftarrow{T_1}$ is a monotonic, ω-continuous, operator;
from lemma 12, $R(\llbracket f \rrbracket_t)$ is directed;
from lemma 20,

$$\bigsqcup_{f \in R(\llbracket f \rrbracket_t .env)} \left(\overleftarrow{T_1} f\right) = \overleftarrow{T_1}\left(\bigsqcup_{f \in R(\llbracket f \rrbracket_t .env)} f\right).$$

It follows that (using the λ-notation for functions),

$$\begin{aligned}
&\overleftarrow{T_1}.E_+([\![f]\!]_t).t_0 \\
&= \sup_{\substack{(U_n)_{n\geq 2} \\ U_n \text{ not depending on } t_0}} \iint \lambda\langle t_2, \ldots\rangle.([\![f]\!]_t\,.env)(\langle t_1, \ldots\rangle) \\
&\qquad\qquad \mathrm{d}[G(T,(U_n)_{n\geq 2}).t_1]\, T_1(t_0, \mathrm{d}t_1) \\
&= \sup_{\substack{(U_n)_{n\geq 2} \\ U_n \text{ not depending on } t_0}} \int \lambda\langle t_2, \ldots\rangle.([\![f]\!]_t\,.env)(\langle t_1, \ldots\rangle) \\
&\qquad\qquad \mathrm{d}[G(T,(U_n)_{n\geq 1}).t_0]
\end{aligned}$$

Let us apply lemma 10 to that last expression. We obtain

$$\overleftarrow{T_1}.E_+([\![f]\!]_t).t_0 = \sup_{(U_n)_{n\geq 2}} \int \lambda\langle t_2, \ldots\rangle.([\![f]\!]_t\,.env)(\langle t_1, \ldots\rangle)\, \mathrm{d}[G(T,(U_n)_{n\geq 1}).t_0] \quad (34)$$

Let $t_0 \in X$. Let us now consider all U_1's.

$$\begin{aligned}
&E_+([\![\mathrm{shift}(f)]\!]_t\,.env).t_0 \\
&= \sup_{(U_n)_{n\geq 1}} \int \lambda\langle t_1, \ldots\rangle.([\![f]\!]_t\,.env) \circ \mathrm{shift}(\langle t_0, t_1, \ldots\rangle) \\
&\qquad\qquad \mathrm{d}[G(T,(U_n)_{n\geq 1}).t_0] \\
&= \sup_{U_1} \sup_{(U_n)_{n\geq 2}} \int \lambda\langle t_1, \ldots\rangle.([\![f]\!]_t\,.env)(\langle t_1, \ldots\rangle) \\
&\qquad\qquad \mathrm{d}[G(T,(U_n)_{n\geq 1}).t_0] \\
&= \left(\sup_{U_1} \overleftarrow{\left(T \circ \left[\begin{smallmatrix}\mathrm{Id}\\ U_1\end{smallmatrix}\right]\right)}.E_+([\![f]\!]_t)\right).t_0 \quad \text{(using Equ. 34)} \\
&= [\![\mathrm{shift}(f)]\!]_{e+}\,.E_+(env)
\end{aligned}$$

- $[\![\mathrm{lfp}(name \mapsto f)]\!]_{e+}\,.env = \mathrm{lfp}(\lambda\phi.[\![f]\!]_{e+}\,.env[name \mapsto \phi])$.
From lemma 9, $\lambda\phi.[\![f]\!]_{e+}\,.env[name \mapsto \phi]$ is ω-upper-continuous.
Also, $[\![\mathrm{lfp}(name \mapsto f)]\!]_t\,.env = \mathrm{lfp}(\lambda\phi.[\![f]\!]_t\,.env[name \mapsto \phi])$.
From lemma 7, $\lambda\phi.[\![f]\!]_t\,.env[name \mapsto \phi]$ is ω-upper-continuous.
From the induction hypothesis,
$E_+ \circ (\lambda\phi.[\![f]\!]_t\,.env[name \mapsto \phi]) = (\lambda\phi.[\![f]\!]_{e+}\,.E_+(env)[name \mapsto \phi]))$.
From lemma 6, E_+ is ω-upper-continuous. The conclusion then follows from lemma 23.

□

The following theorem guarantees the soundness of the abstract analysis for all formulas.

Theorem 14. *Let f be a formula. Let env be an environment of valuators. Let us suppose that $H \geq E_+(env)$ pointwise. Then*

$$[\![f]\!]_{e+}.(H) \geq E_+([\![f]\!]_t.env). \tag{35}$$

The proof is similar as that of Th. 13.

Also similarly we can guarantee the soundness of the analysis of summations:

Theorem 15. *The semantics of the summing operator satisfies:*

$$E_+([\![\Sigma f]\!]_t) = [\![\Sigma f]\!]_{e+}. \tag{36}$$

6 Abstract Analysis

We shall see here more precisely how to apply abstract interpretation to that backwards semantics.

6.1 General case

We compute safe approximations of $[\![f]\!]_{e+}$ by abstract interpretation. We introduce an abstract semantics $[\![f]\!]^\sharp_{e+}$ which is an upper approximation of f:

$$\forall env\, \forall env^\sharp\ env^\sharp \geq env \implies [\![f]\!]^\sharp_{e+}.env^\sharp \geq [\![f]\!]_{e+}.env. \tag{37}$$

The computations for $[\![f]\!]^\sharp_{e+}$ will be done symbolically in an *abstract domain* such as the ones described in [23, 24] (Fig. 5 and 6).

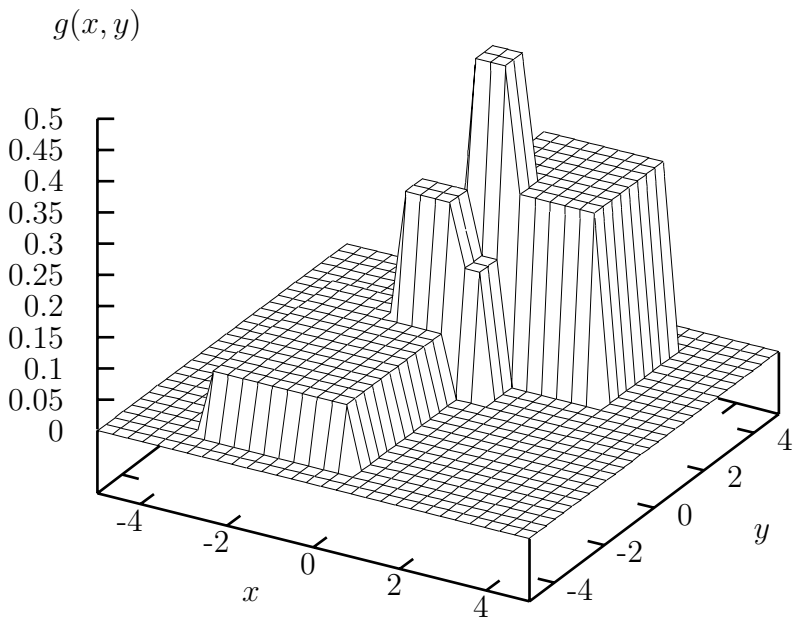


Fig. 5. An abstract domain: step functions (linear combinations of characteristic functions of elements of another set, here the set of products of intervals. [23]

- We shall assume that we have an abstract operation for "shift". That is, we have a monotone operation $\mathrm{pre}^\sharp$ such that

$$\forall f,\ \forall T \in \mathcal{T},\ \mathrm{pre}^\sharp.f^\sharp \geq T^*.f^\sharp. \tag{38}$$

  This operation will be supplied by the abstract domain that we use. Then

$$\forall env,\ \forall env^\sharp,\ env^\sharp \geq env \implies \\ [\![\mathrm{shift}(f)]\!]^\sharp_{e+}.env^\sharp \geq [\![\mathrm{shift}(f)]\!]_{e+}.env. \tag{39}$$

  provided that

$$\forall env,\ \forall env^\sharp,\ env^\sharp \geq env \implies [\![f]\!]^\sharp_{e+}.env^\sharp \geq [\![f]\!]_{e+}.env.$$

- We shall approximate least fixpoints using a *widening operator* [8, §4.3]. A widening operator $\triangledown$ is a kind of abstraction of the least upper bound that enforces convergence:

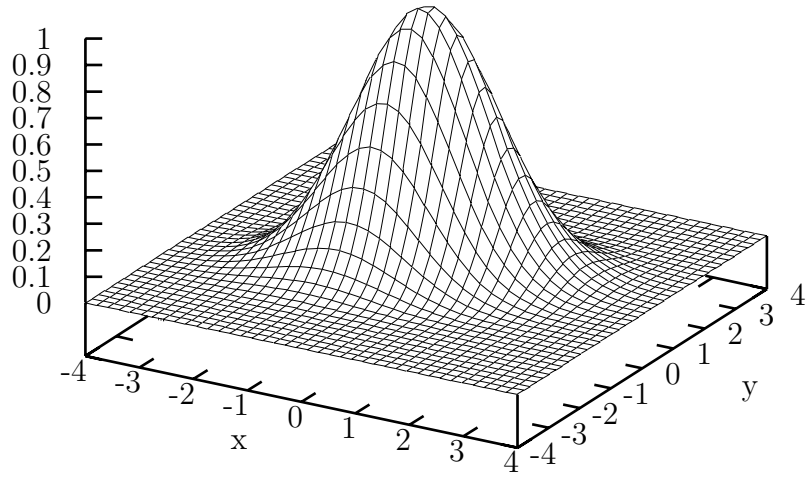


Fig. 6. An abstract domain: normal (Gaussian) distribution densities. [24]

- $f \nabla g \geq \sup(f, g)$ (pointwise);
- For any ascending sequence $(v_n)_{n\in\mathbb{N}}$, the sequence $(u_n)_{n\in\mathbb{N}}$ defined inductively by $u_{n+1} = u_n \nabla v_n$ is to be ultimately stationary.

Then the limit $L^\sharp$ of the sequence defined by $u_0 = 0$ and $u_{n+1} = u_n \nabla f^\sharp(u_n)$, where $f^\sharp$ is an upper approximation of f, is an upper approximation to the least fixpoint of f. More precise upper approximations of the least fixpoint of f can then be reached by iterating $f^\sharp$ over $L^\sharp$ using a so-called *narrowing operators* [8, §4.3].

- We shall approximate greatest fixpoints using a limited iteration approach: if $f^\sharp$ is an upper approximation of f, then for any $n \in \mathbb{N}$, $f^{\sharp n}(\top) \geq \mathrm{gfp}\, f$.

6.2 Partitioning in programs

In the case of programs, the state space is generally $P \times M$, where P is the (finite) set of program points and M the set of possible

memory configurations (as in Fig. 4). More generally, P may be a kind of *partitioning* of the program. Non-probabilistic analysis generally operates on abstractions of $\mathcal{P}(P\times M) \simeq P\times M \to \{0,1\} \simeq P \to \mathcal{P}(M)$. Given an abstraction of $\mathcal{P}(M)$ by a lattice $L^\sharp$, one obtains a pointwise abstraction of $P \to \mathcal{P}(M)$ by $P \to L^\sharp$. Elements of $P \to L^\sharp$ are just vectors of $|P|$ elements of $L^\sharp$.

For instance, in the case of the program in Fig. 4, one would partition according to Π, and may, for instance, use a domain of piecewise polynomial functions for an exact abstraction, or a domain of piecewise linear function for an approximate abstraction.

This approach can be directly extended to our measurable functions: we shall abstract $P \times M \to I$ (where $I = [0,1]$ or $I = [0,+\infty]$) by $P \to L^\sharp$ if $L^\sharp$ is an abstract domain for $M \to I$.

The first problem is to get an abstraction of the operation used in the "shift" construct:

$$F : \left| \begin{array}{l} (P \times M \to I) \to (P \times M \to I) \\ h \qquad\qquad\quad \mapsto (l,m) \mapsto \sup_{y\in Y} \\ \qquad\qquad\qquad \Sigma_{(l',m')\in P\times M}\, T\left((l,m),y;(l',m')\right).h(l',m') \end{array} \right. \tag{40}$$

Let us take the following form for the program instructions: at program point l, the executed instruction represented by T is the sequence:

(1) a nondeterministic choice y is taken in the set Y_l;
(2) a random choice r is taken in set R_l according to distribution $\mathcal{R}_p$;
(3) the memory state is combined deterministically with the two choices to form the new memory state using a function $F_l : (M \times Y) \times R_l \to M$;
(4) depending on the memory state m, the program takes a deterministic jump to program point $J(l,m)$.

Let us note $\tau_l(l') = \{m \mid J(l,m) = l'\}$ (the set of memory values m that lead to program point l' from program point l; $\tau_l(l')$ is then essentially the condition for a conditional jump). Then we can rewrite the transition equation as follows

$$(F.h)(l) = \mathrm{choice}^*_{Y_l} \circ \mathrm{random}^*_{R_l} \circ (F_l)^*_p \left(\sum_{l' \in P} \phi^*_{\tau_l l'} \left(h(l', \bullet) \right) \right) \quad (41)$$

using the following building blocks:

$$\mathrm{choice}^*_{Y_l}(h) = m \mapsto \sup_{y \in Y_l} h(m,y) \quad (42)$$

$$\mathrm{random}^*_{R_l}(h) = m \mapsto \int h(m,r)\ \mathrm{d}\mu_{\mathcal{R}_l}(r) \quad (43)$$

$$(F_l)^*_p(h) = h \circ F_l \quad (44)$$

$$\phi^*_A(h) = h.\chi_A \quad (45)$$

The reasons for those notations are explained in earlier works on the linear adjoint of Kozen's denotational semantics for probabilistic programs [23].

We shall abstract F as the composition of abstractions for:

- $\mathrm{choice}^*_{Y_l}$, nondeterministic choice;
- $\mathrm{random}^*_{R_l}$, probabilistic choice;
- $F^*_{l_p}$, deterministic run (arithmetic operations and the like);
- ϕ^*_A, test.

Since F is a composition of ω-upper-continuous functions (Def. 16, Lemma 5), it is also ω-upper-continuous. Therefore he least fixpoint of F is obtained as the limit of $F^n(0)$ (Lemma 22). This is the point-wise limit of a sequence of functions from Ω to I. The expression of the iterates using a partition with respect to P is as follows:

$$f_1^{(n+1)} = F_1(f_1^{(n)}, \ldots, f_{|P|}^{(n)}) \quad (46)$$

$$\vdots\!\vdots \quad (47)$$

$$f_{|P|}^{(n+1)} = F_{|P|}(f_1^{(n)}, \ldots, f_{|P|}^{(n)}) \quad (48)$$

$$(49)$$

In terms of implementation, this means that we update in parallel the $|P|$ elements of the vector representing the iterate. As noted by Cousot [7, §2.9], this parallel update may be replaced by *chaotic iterations* or *asynchronous iterations*. Chaotic iterations allow us to compute the iterations by taking into account the recently updated elements. All these iteration strategies lead to the same limit (the least fixpoint of F).

Let us consider for instance the following strategy:

$$\begin{array}{l} f_1^{(n+1)} = F_1(f_1^{(n)}, \ldots, f_{|P|}^{(n)}) \\ f_2^{(n+1)} = F_2(f_1^{(n+1)}, \ldots, f_{|P|}^{(n)}) \\ \vdots \qquad \vdots \\ f_{|P|}^{(n+1)} = F_{|P|}(f_1^{(n+1)}, \ldots, f_{|P|}^{(n)}) \end{array} \tag{50}$$

This strategy is itself a monotone operator whose least fixpoint is to be determined. It has an obvious abstract counterpart leading to an approximate fixpoint in the usual way (§6.1).

7 Conclusions and Discussion

We showed how to apply abstract interpretation techniques to check various temporal properties of (nondeterministic) probabilistic programs, considered as Markov decision processes (small-step semantics).

The most natural point of view on those processes is that the nondeterministic decisions are taken as the program proceeds, taking into account the current state as well as the previous ones. This how Markov decision processes are usually studied [28] and this is the approach we took here.

It can be argued that this model is excessively pessimistic. Indeed, if nondeterminism is used to model the environment of an embedded system, then it is excessive to assume that the behavior

of this environment depends on the history of the *internal state* of the system; only the part of this history observable from the environment should be taken into account. This leads to the study of *partially observable Markov decision processes* (POMDP); however, their effective analysis is much more complex than that of fully observable processes [17].

Cleaveland's work [5] focuses on the model where the nondeterministic choices are taken *after* the probabilistic ones. This simplifies the theory to some extent, since taking the product of the analyzed process with an nondeterministic "observation" process, such as a nondeterministic Büchi automaton, is then easy. We have already proposed a Monte-Carlo method for such semantics [21].

The backwards analysis method we described is a generalization of the value iteration method used in operational research to compute the value of Markov decision processes. Our reachability analysis is related to the study of *positive bounded models* [28, §7.2], where the reward 1 is granted the first time the process runs through the set of states to consider. The liveness analysis is related to the study of *negative models* [28, §7.3], where the reward -1 is granted the first time the process leaves the set of states to consider.

In dealing with non-probabilistic programs, flow-sensitive analyses are generally defined in a *forward* fashion. Similarly, early definitions of the semantics of probabilistic programs were described as extensions of conventional denotational semantics [13,14]. The concrete domain is then the set of probability distributions. However, such semantics restrict programs to be probabilistically deterministic, i.e. the distribution of the next state is known given the current state of the program (Markov-chain semantics); there is no provision for nondeterministic choice in the sense of arbitrary choice with no known statistical data. Forward semantics were later extended to nondeterministic choice using *convex sets* of probability distributions [25]. We proposed program analysis

methods based on sets of probability distributions [19]. However, one significant problem that arises is the complexity of representing sets (even if only convex) of probability distributions (themselves complex objects) over the state space X. In contrast, working in a backward fashion just needs representing functions from X to $[0, 1]$.

Naive representations of sets of distributions may yield very imprecise results. For instance, it can be very imprecise to represent a set of distributions μ on a finite or countable state space X with a function $f : X \to [0, 1]$ as $\{\mu \mid \forall x \in X,\ \mu(\{x\}) \leq f(x)$ in the presence of nondeterministic choices. For instance, in the program

```
if (nondeterministic_choice) {
  /* A */
} else {
  /* B */
}
/* C */
```

$f(\mathtt{A}) = f(\mathtt{B})$, but then the rules for program meet points [19, §2.2] yield $f(\mathtt{C}) = 2$, which is of course a correct upper bound for the maximal probability of reaching C, but is still very imprecise. Worse, since abstraction is somewhat equivalent to introducing supplemental nondeterminism, abstraction may lead to overestimates, depending on its precision [19, §5.2]. For these reasons, we preferred a backward analysis. However, we also presented a forward analysis based on the same logic for trace sets [22, §8.4]. How to combine forward and backwards analyses satisfactorily is still an open problem.

Formal languages similar to the one we consider have been introduced by other authors, such as *quantitative game μ-calculus* [11]. The differences between our approach and this game calculus approach are threefold:

- We give a semantics in terms of traces, then prove its link with a semantics in terms of expectation functions; quantitative μ-

calculus only gives the interpretation as expectation functions.
- While we prove a generic link between the semantics as an inequality valid for any formula (or an equation for some class), de Alfaro proves an equality for some specific formulas (reachability, liveness, deterministic Büchi and Rabin trace properties). We conjecture that we can extend the equality cases of this link.
- De Alfaro considers random two-player games while we consider random single-player games. We mentioned briefly (§5.2) the differences between Markov decision processes and two-player games. Such problems can model questions such as the choice of an optimal strategy by the program so as to minimize the probability of a problem for all possible environments.

A possible extension of these properties is *discounted models* [28, Ch. 6]. In these, the importance of the future decreases exponentially; for instance, λ-discounted reachability would count passing through A for the first time at step n as λ^n instead of 1 (of course, $0 < \lambda < 1$). The theoretical study of those models is considerably easier than that of non-discounted models, since the fixpoints to study are the fixed points of contraction mappings in Banach spaces. While the extension of the techniques exposed in this paper to discounted models is easy (it suffices to add a multiplication by λ in the semantics of the "shift" operation), the practical interest of such models in the checking of computer programs remains to be seen.

Another possible extension is the computation of averages not only on the space of the program, but also on the time: computing the average value of a certain function as long as the program is running. Since this property is the quotient of two summing properties, there is no obvious method to evaluate it iteratively.

Another possible direction is the study of continuous time probabilistic systems. As usual with continuous-time systems, some kind of reduction to discrete time processes is to be done [15, 16].

References

[1] Christel Baier, Edmund M. Clarke, Vasiliki Hartonas-Garmhausen, and Marta Kwiatkowska. Symbolic model checking for probabilistic processes. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Automata, Languages and Programming (ICALP '97)*, volume 1256 of *LNCS*, pages 430–440. Springer, 1997.

[2] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *FST TCS 95: Foundations of Software Technology and Theoretical Computer Science*, volume 1026 of *Lecture Notes in Computer Science*, pages 499–513. Springer-Verlag, 1995.

[3] C.Baier, M.Kwiatkowska, and G.Norman. Computing probability bounds for linear time formulas over concurrent probabilistic systems. *Electronic Notes in Theoretical Computer Science*, 21, 1999.

[4] Edmund M. Clarke, Jr, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 1999.

[5] Rance Cleaveland, Scott A. Smolka, and Amy E. Zwarico. Testing preorders for probabilistic processes. In Werner Kuich, editor, *Automata, Languages and Programming, 19th International Colloquium*, volume 623 of *Lecture Notes in Computer Science*, pages 708–719, Vienna, Austria, 13–17 July 1992. Springer-Verlag.

[6] C. Courcoubetis and M. Yannakakis. Markov decision processes and regular events. In *Proc. ICALP'90*, volume 443 of *LNCS*, pages 336–349. Springer, 1990.

[7] Patrick Cousot. *Méthodes itératives de construction et d'approximation de points fixes d'opérateurs monotones sur un treillis, analyse sémantique de programmes*. Thèse d'état ès sciences mathématiques, Université scientifique et médicale de Grenoble, Grenoble, France, 21 mars 1978.

[8] Patrick Cousot and Radhia Cousot. Abstract interpretation and application to logic programs. *J. Logic Prog.*, 2-3(13):103–179, 1992.

[9] Pedro d'Argenio, Bertrand Jeannet, Henrik Jensen, and Kim Guldstrand Larsen. Reduction and refinement strategies for probabilistic analysis. In Holger Hermanns and Roberto Segala, editors, *Process Algebra and Probabilistic Methods : Performance Modeling and Verification*, volume 2399 of *LNCS*, pages 57–76. Springer, July 25–26 2002.

[10] Luca de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, Department of Computer Science, June 1998. CS-TR-98-1601.

[11] Luca de Alfaro and Rupak Majumdar. Quantitative solution of omega-regular games. In *STOC'01, 33rd Annual ACM Symposium on Theory of Computing*, pages 675–683. ACM, 2001.

[12] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reability. Technical Report R90-13, Swedish Institute of Computer Science, December 1990.

[13] D. Kozen. Semantics of probabilistic programs. In *20th Annual Symposium on Foundations of Computer Science*, pages 101–114, Long Beach, Ca., USA, October 1979. IEEE Computer Society Press.

[14] D. Kozen. Semantics of probabilistic programs. *Journal of Computer and System Sciences*, 22(3):328–350, 1981.

[15] Marta Z. Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. Verifying quantitative properties of continuous probabilistic timed automata. Technical Report CSR-00-6, University of Birmingham, School of Computer Science, March 2000.

[16] Marta Z. Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. Verifying quantitative properties of continuous probabilistic timed automata. In C. Palamidessi, editor, *CONCUR 2000 - Concurrency Theory 11th International Conference*, number 1877 in LNCS, pages 123–137. Springer, 2000.

[17] Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Efficient dynamic-programming updates in partially observable markov decision processes. Technical Report CS-95-19, Brown University, 1995.

[18] A. McIver. Reasoning about efficiency within a probabilistic μ-calculus. In *Proc. of PROBMIV*, pages 45–58, 1998. Technical Report CSR-98-4, University of Birmingham, School of Computer Science.

[19] David Monniaux. Abstract interpretation of probabilistic semantics. In *Seventh International Static Analysis Symposium (SAS'00)*, number 1824 in Lecture Notes in Computer Science, pages 322–339. Springer Verlag, 2000. Extended version on the author's web site.

[20] David Monniaux. An abstract analysis of the probabilistic termination of programs. In *8th International Static Analysis Symposium (SAS'01)*, number 2126 in Lecture Notes in Computer Science, pages 111–126. Springer Verlag, 2001.

[21] David Monniaux. An abstract Monte-Carlo method for the analysis of probabilistic programs (extended abstract). In *28th Symposium on Principles of Programming Languages (POPL '01)*, pages 93–101. Association for Computer Machinery, 2001.

[22] David Monniaux. *Analyse de programmes probabilistes par interprétation abstraite*. Thèse de doctorat, Université Paris IX Dauphine, 2001. Résumé étendu en français. Contents in English.

[23] David Monniaux. Backwards abstract interpretation of probabilistic programs. In *European Symposium on Programming Languages and Systems (ESOP '01)*, number 2028 in Lecture Notes in Computer Science, pages 367–382. Springer Verlag, 2001.

[24] David Monniaux. Abstraction of expectation functions using gaussian distributions. In Lenore D. Zuck, Paul C. Attie, Agostino Cortesi, and Supratik Mukhopadhyay, editors, *Verification, Model Checking, and Abstract Interpretation: VMCAI '03*, number 2575 in Lecture Notes in Computer Science, pages 161–173. Springer Verlag, 2003.

[25] Carroll Morgan, Annabelle McIver, Karen Seidel, and J. W. Sanders. Probabilistic predicate transformers. Technical Report TR-4-95, Oxford University, February 1995.

[26] Jacques Neveu. *Mathematical Foundations of the Calculus of Probabilities.* Holden-Day, 1965.

[27] Jacques Neveu. *Bases mathématiques du calcul des probabilités.* Masson et Cie, Éditeurs, Paris, 1970. Préface de R. Fortet. Deuxième édition, revue et corrigée.

[28] Martin L. Puterman. *Markov decision processes: discrete stochastic dynamic programming.* Wiley series in probability and mathematical statistics. John Wiley & Sons, 1994.

[29] Walter Rudin. *Real and Complex Analysis.* McGraw-Hill, 1966.

[30] Roberto Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems.* PhD thesis, Massachusetts Institute of Technology, 1995. Technical report MIT/LCS/TR-676.

[31] W. Thomas. Automata on infinite objects. In van Leeuwen [32], pages 135–191.

[32] J. van Leeuwen, editor. *Handbook of Theoretical Computer Science, vol. B.* Elsevier, 1990.

A Theory of ordered sets

Definition 16. Let $(X, \sqsubseteq_X)$ and $(Y, \sqsubseteq_Y)$ be two sets. $f : X \to Y$ is a *monotone* operator if for all $a, b \in X$, $a \sqsubseteq_X b \implies f(a) \sqsubseteq_Y f(b)$.

f is a *upper-continuous* if for all ascending sequence $(x_i)_{i \in I}$ of elements of X such that the least upper bound $\bigsqcup_{i \in I} x_i$ is defined, then $f\left(\bigsqcup_{i \in I} x_i\right) = \bigsqcup_{i \in I} f(x_i)$ (respectively for *lower-continuous*, descending sequences, and greatest lower bounds instead of least upper bounds). f is *continuous* if it is both upper- and lower-continuous.

The same, restricted to countable sequences, defines *ω-upper-continuous*, *ω-lower-continuous* and *ω-continuous*.

Lemma 17. *For any monotone operator $\phi : X \to Y$ and subset K of X, $\phi(\sqcup K) \sqsupseteq \sqcup_{f \in K} \phi(f)$.*

Proof. For any $f \in K$, $f \sqsubseteq \sqcup K$ and thus $\phi(f) \sqsubseteq \phi(\sqcup K)$. The result follows. □

Definition 18. An ordered set $(L, \sqsubseteq)$ is call *directed* if for all x and y in L there exists z such that both $x \sqsubseteq z$ and $y \sqsubseteq z$.

Lemma 19. *Let X be a finite or countable set of states. Let K be a directed (Def. 18) subset of $X \to I$. Then there exists an ascending sequence of elements of K that converges point-wise to $\sqcup K$, the point-wise upper-bound of K.*

Proof. We shall assimilate X to either the finite set $\{0, \ldots, N-1\}$ or $\mathbb{N}$ (in which case $N = \infty$), depending on its cardinality. Let $F = \sqcup K$.

Let us construct such a sequence $(f_n)_{n \in \mathbb{N}^*}$ of elements of K that converges point-wise to F. Let $n \in \mathbb{N}$. Let us construct by recurrence an ascending sequence $(g_k)_{1 \leq k \leq \min(n, N-1)}$ of elements of K:

$n = 1$ There exists a $g_1 \in K$ such that $g_1(0) \geq F(0) - 1/n$ if $F(0) < \infty$ or $g_1(0) \geq n$ if $F(0) = +\infty$.

$n > 1$ There exists a $g \in K$ such that $g(k) \geq F(k) - 1/n$ if $F(k) < \infty$ or $g(k) \geq n$ if $F(0) = +\infty$; since g and g_{k-1} belong to K and K is directed, we can therefore take $g_k \in K$ such that $g_k \geq g$ and $g_k \geq g_{k-1}$.

Let f_n be g_n.

Let us now construct by recurrence an ascending sequence $(\tilde{f}_n)_{n \in \mathbb{N}^*}$ of elements of K:

$n = 1$ Let $\tilde{f}_1 = f_1$.

$n > 1$ There exists a $\tilde{f}_n \in K$ such that $\tilde{n}_n \geq f_n$ and $\tilde{f}_n \geq f_{n-1}$.

Let us show that $(\tilde{f}_n)_{n \in \mathbb{N}^*}$ converges point-wise to F. Let $x \in X$.

- Case when $F(x) < \infty$. If $n \geq x$, then $f_n(x) \geq F(x) - 1/n$. Thus $F(x) - 1/n \leq \tilde{f}_n(x) \leq F(x)$ and $\lim n \to \infty \tilde{f}_n(x) = F(x)$.
- Case when $F(x) = \infty$. If $n \geq x$, then $f_n(x) \geq n$, thus $\lim_{n\to\infty} \tilde{f}_n(x) = F(x)$.

□

Lemma 20. *Let Y be an ordered set. Let $\phi : (X \to I) \to Y$ be a monotonic, ω-upper-continuous function. Let K be a directed subset of $X \to I$. Then $\phi(\sqcup K) = \bigsqcup_{f\in K} \phi(f)$.*

Proof. From lemma 17, $\phi(\sqcup K) \sqsupseteq \bigsqcup_{f\in K} \phi(f)$.

K is directed. From lemma 19, there exists an ascending sequence f_n such that $\sqcup_n f_n = F$. Since ϕ is ω-upper-continuous, $\bigsqcup_n \phi(f_n) = \phi(\sqcup K)$. But $\bigsqcup_n \phi(f_n) \sqsubseteq \bigsqcup_{f\in K} \phi(f)$ since the f_n belong to K. Therefore $\phi(\sqcup K) \sqsubseteq \bigsqcup_{f\in K} \phi(f)$. □

Lemma 21. *The least upper bound of a set of ω-upper-continous functions is ω-upper-continous.*

Lemma 22. *Let T_1 and T_2 be two complete lattices. Let $\psi : T_1 \times T_2 \to T_1$ be an ω-upper-continuous operator. Then $y \mapsto \mathrm{lfp}(x \mapsto \psi(x, y))$ is an ω-upper-continuous operator.*

Proof. $y \mapsto \mathrm{lfp}(x \mapsto \psi(x, y))$ is ω-upper-continuous and thus $\mathrm{lfp}(x \mapsto \psi(x, y)) = \bigsqcup_n (y \mapsto \mathrm{lfp}(x \mapsto \psi(x, y)))^n (\bot)$. The result then follows from lemma 21. □

Lemma 23. *Let T_1 and T_2 be two complete lattices. Let $\alpha : T_1 \to T_2$ be an ω-upper-continuous operator such that $\alpha(\bot) = \bot$. Let $\psi_1 : T_1 \to T_1$ and $\psi_2 : T_2 \to T_2$ be two ω-upper-continuous operators such that $\psi_2 \circ \alpha = \alpha \circ \psi_1$. Then $\alpha(\mathrm{lfp}\, \psi_1) = \mathrm{lfp}\, \psi_2$.*

Proof. $\alpha(\mathrm{lfp}\, \psi_1) = \alpha(\bigsqcup_n \psi_1^n(\bot)) = \bigsqcup_n \alpha \circ \psi_1^n(\bot) = \bigsqcup_n \psi_2^n(\underbrace{\alpha(\bot)}_{\bot}) =$ $\mathrm{lfp}\, \psi_2$. □

B Measure theory and integrals

B.1 Measure theory

We express probabilities using *measures* [29, §1.18]. Measures express the intuitive idea of a "repartition of weight" on the domain; probabilities are a particular case of measures. A measure on a space Ω assigns a "weight" to subsets of Ω. A probability measure is a measure of total weight 1.

Before entering the mathematical definitions, let us see a few familiar examples:

- In the case where Ω is finite or countable, defining a positive measure is just defining a function $f : \Omega \to \mathbb{R}_+$ (the weight of $A \subseteq \Omega$ is then $\sum_{\omega \in A} f(a)$); it is a a probability measure if $\sum_{\omega \in \Omega} f(\omega) = 1$. A measure on a finite space is said to be equidistributed if $\forall \omega \ f(\omega) = \frac{1}{|\Omega|}$.
- In the case of the real field $\mathbb{R}$, the Lebesgue measure μ is so that the measure of a segment $[a, b]$ is its length. The Lebesgue measure on $[0, 1]$ formalizes that familiar notion of a real "equidistributed in $[0, 1]$". The Lebesgue measure can be defined on $\mathbb{R}^n$, and the measure of an object is its area (for $\mathbb{R}^2$) or volume (for $\mathbb{R}^3$).
- The *unit point mass* (or *Dirac measure*) at x is the measure δ_x defined by: $\delta_x(A) = 1$ if $x \in A$, $\delta_x(A) = 0$ otherwise.

Let us see now the formal definitions:

Definition 24. A *σ-algebra* is a set of subsets of a set X that contains $\emptyset$ and is stable by countable union and complementation (and thus contains X and is stable by countable intersection).

In the case of the Lebesgue measure, we shall consider a suitable σ-algebra, such as the Borel or Lebesgue ones [29]. It is sufficient to say that most sets that one can construct are Lebesgue-measurable.

Definition 25. A set X with a σ-algebra σ_X defined on it is called a *measurable space* and the elements of the σ-algebra are the *measurable subsets*.

We shall often mention measurable spaces by their name, omitting the σ-algebra, if no confusion is possible.

Definition 26. If X and Y are measurable spaces, $f : X \to Y$ is a *measurable function* if for all W measurable in Y, $f^{-1}(W)$ is measurable in X.

Definition 27. A *positive measure* is a function μ defined on a σ-algebra σ_X whose range is in $[0, \infty]$ and which is countably additive. μ is countably additive if, taking $(A_n)_{n\in\mathbb{N}}$ a disjoint collection of elements of σ_X, then

$$\mu\left(\bigcup_{n=0}^{\infty} A_n\right) = \sum_{n=0}^{\infty} \mu(A_n). \tag{B.1}$$

To avoid trivialities, we assume $\mu(A) < \infty$ for at least one A. The *total weight* of a measure μ, noted $|\mu|$, is $\mu(X)$. μ is said to be *concentrated* on $A \subseteq X$ if for all B, $\mu(B) = \mu(B \cap A)$. We shall note $\mathcal{M}_+(X)$ the positive measures on X.

Definition 28. A *σ-finite measure* on X is a measure μ such that there exists a countable family of measurable sets $(A_n)_{n\in\mathbb{N}}$ such that $\forall n\ \mu(A_n) < \infty$ and $\cup_n A_n = X$. We note by $\mathcal{M}_+(X)$ the σ-finite measures on X.

Definition 29. A *probability measure* is a positive measure of total weight 1; a *sub-probability measure* has total weight less or equal to 1. We shall note $\mathcal{M}_{\leq 1}(X)$ the sub-probability measures on X.

Definition 30. Given two sub-probability measures μ and μ' (or more generally, two σ-finite measures) on X and X' respectively, we note $\mu \otimes \mu'$ the product measure [29, definition 7.7], defined on the product σ-algebra $\sigma_X \times \sigma_{X'}$. The characterizing property of this product measure is that $\mu \otimes \mu'(A \times A') = \mu(A).\mu'(A')$ for

all measurable sets A and A'.

These definitions constitute the basis of the theory of Lebesgue integration [29, ch. 1, 2], one of the most essential results of which is:

Theorem 31 (Lebesgue's monotone convergence theorem). *Let (X, μ) be a measured space. Let f_n be an ascending sequence of functions, whose point-wise limit is f. Then the sequence $\int f_n \, \mathrm{d}\mu$ is also ascending and $\lim_{n\to\infty} \int f_n \, \mathrm{d}\mu = \int f \, \mathrm{d}\mu$.*

B.2 Construction of measures on infinite sequences with transition probabilities

The intuitive meaning of this theorem [26, 27, proposition V-I-1] is as follows: if $(E_t)_{t\in\mathbb{N}}$ is a sequence of measurable spaces and the $(P_{t+1}^{0,\ldots,t})_{t\in\mathbb{N}}$ is a sequence of transition probabilities, respectively from $E_0 \times E_t$ to E_{t+1}, then we can construct a transition probability P from E_0 to $E_1 \times E_2 \times \cdots$ such that for each $x_0 \in E_0$, $P(x_0, \cdot)$ is the probability distribution on traces starting from x_0 and following the transition probabilities $(E_t)_{t\in\mathbb{N}}$.

In an even more intuitive fashion: "knowing the starting probability measure, and the transition probabilities to the next states, we can construct the corresponding probability measure on infinite traces".

Theorem 32 (Ionescu Tulcea). *Let $(E_t, \mathcal{F}_t)_{t\in\mathbb{N}}$ be an infinite sequence of measurable spaces and, for any $t \in \mathbb{N}$, let $P_{t+1}^{0,\ldots,t}$ be a transition probability relative to the spaces $(\prod_{s=0}^{t} E_s, \bigotimes_{s=0}^{t} \mathcal{F}_s)$ and $(E_{t+1}, \mathcal{F}_{t+1})$. Then there exists for any $x_0 \in E_0$ a unique probability P_{x_0} on*

$$(\Omega, \mathcal{A}) = \prod_t (E_t, \mathcal{F}_t)$$

whose value for all measurable Cartesian product $\prod_t F_t$ is given

by:

$$P_{x_0}\left[\prod_t F_t\right] = \chi_{A_0}(x_0)\int_{x_1\in F_1} P_1^0(x_0;\mathrm{d}x_1)\int_{x_2\in F_2} P_2^{0,1}(x_0,x_1;\mathrm{d}x_2) \cdots \int_{x_T\in F_T} P_T^{0,\ldots,T-1}(x_0,\ldots,x_{T-1};\mathrm{d}x_T) \quad \text{(B.2)}$$

as long as T is sufficiently great such that $F_t = E_t$ if $t > T$ (the second member is then independent of the chosen T). For any positive random variable Y on $(\Omega,\mathcal{A})$ only depending on the coordinates up to T, we have:

$$\int_\Omega Y(\omega')P_{x_0}(\mathrm{d}\omega') = \int_{F_1} P_1^0(x_0;\mathrm{d}x_1)\int_{F_2} P_2^{0,1}(x_0,x_1;\mathrm{d}x_2) \cdot \int_{x_T\in F_T} Y(x_0,\ldots,x_T)P_T^{0,\ldots,T-1}(x_0,\ldots,x_{T-1};\mathrm{d}x_T) \quad \text{(B.3)}$$

Furthermore, for any positive random variable Y on $(\Omega,\mathcal{A})$,

$$x_0 \mapsto \int Y(\omega')P_{x_0}(\mathrm{d}\omega') \quad \text{(B.4)}$$

is a positive random variable on $(E_0,\mathcal{F}_0)$.