Abstract interpretation of programs as Markov decision processes

David Monniaux

http://www.di.ens.fr/~monniaux
École Normale Supérieure
Département d'Informatique
45, rue d'Ulm
75230 Paris cedex 5
France

Abstract. We propose a formal language for the specification of trace properties of probabilistic, nondeterministic transition systems, encompassing the properties expressible in Linear Time Logic. Those formulas are in general undecidable on infinite deterministic transition systems and thus on infinite Markov decision processes. This language has both a semantics in terms of sets of traces, as well as another semantics in terms of measurable functions; we give and prove theorems linking the two semantics. We then apply abstract interpretation-based techniques to give upper bounds on the worst-case probability of the studied property. We propose an enhancement of this technique when the state space is partitioned — for instance along the program points —, allowing the use of faster iteration methods.

1 Introduction

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.

The analysis of *finite-state* Markov decision processes was originally conducted in the fields of operational research and finance mathematics [23]. More recently, they have been studied from the angle of probabilistic computing systems [1–4,15,24]. Effective resolution techniques include linear programming [23, §7.2.7] [7] and newer data structures such as MTBDDs [2]. However, the problem of large- or infinite-state systems has not been so well studied.

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* [9], a general theory of approximation between semantics.

We have earlier proposed two classes of automatic methods to analyze such system: some *forward* [16,17], some *backward* [19,20]. 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 [8, §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.

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 process*.

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.

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 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: $\overrightarrow{T}(d)(\omega_2) = \sum_{\omega_1\in\Omega_1} T(\omega_1,\omega_2)d(\omega_1)$.*

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 its 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*.[1]

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: $\overleftarrow{T}(f)(\omega_1) = \sum_{\omega_2 \in \Omega_2} T(\omega_1, \omega_2) f(\omega_2)$.*

Lemma 1. *For all transition probability T, $\overleftarrow{T}$ is ω-continuous.*

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

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

2.2 Probability measures on traces

We shall also 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.

We use the theorem of Ionescu Tulcea (Appendix B) 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* [23] , which has been studied more particularly in the field of operational research and finance mathematics, as well as machine learning.

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.

[1] 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.

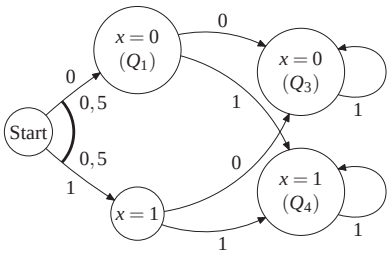
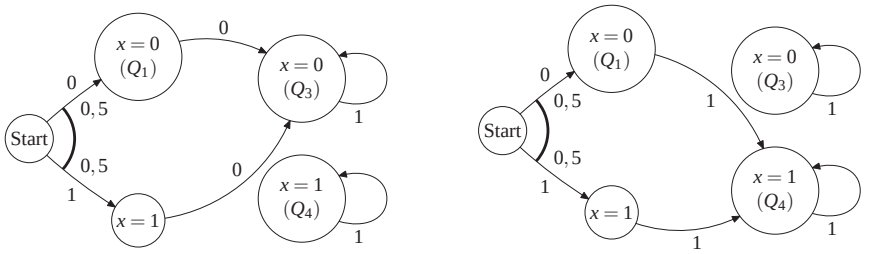


Fig. 1. A probabilistic-nondeterministic transition system



(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.

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* [23, §2.1.5].

In this paper, we shall assume that the adversary may see the present and past states of the execution. This is realistic if the adversary models a human evildoer as well as the physical environment of an embedded system.

Let us note that other choices 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 of x and y (Fig. 1). Clearly, if the nondeterminism 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, 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,

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

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

$$S_T(f, (U_n)_{n\in\mathbb{N}}) = t_0 \mapsto \langle \boldsymbol{t} \mapsto f(t_0, \boldsymbol{t}), G(T, (U_n)_{n\in\mathbb{N}})(t_0)\rangle \tag{3}$$

(S short for S_T if there is no ambiguity about T) and $R(f)$ the set of all functions $S(T, (U_n)_{n\in\mathbb{N}})$ when $(U_n)_{n\in\mathbb{N}}$ is a sequence of transition probabilities, U_n being a transition probability from Ω^n to Y. The 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 [23]).

Lemma 5 *$E_+(1) = 1$ 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".

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

We shall consider formulas written in the following language:

$$
\begin{aligned}
\mathit{formula1} ::= {} & \mathit{name} \\
& \mid \mathit{constant} \\
& \mid \mathit{name} +_{\mathit{set}} \mathit{formula1} \text{ where } \mathit{set} \subseteq \Omega \\
& \mid \mathit{constant} +_{\mathit{set}} \mathit{formula1} \\
& \mid \mathrm{lfp}(\mathit{name} \mapsto \mathit{formula1}) \\
& \mid \mathrm{gfp}(\mathit{name} \mapsto \mathit{formula1}) \\
& \mid \mathrm{shift}(\mathit{formula1}) \\
& \mid \text{let } \mathit{name} = \mathit{formula1} \text{ in } \mathit{formula2}
\end{aligned}
$$

Let $\mathrm{shift} : \Omega^{\mathbb{N}} \to \Omega^{\mathbb{N}}$: $(\mathrm{shift}.t)_k = t_{k+1}$.

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:

$$
\begin{aligned}
[\![\mathit{name}]\!]_t.\mathit{env} &= \mathit{env}(\mathit{name}) && (4)\\
[\![\mathit{constant}]\!]_t.\mathit{env} &= \mathit{constant} && (5)\\
[\![f_1 +_S f_2]\!]_t.\mathit{env} &= \lambda \boldsymbol{t}.\chi_S(t_0).([\![f_1]\!]_t\ \mathit{env}) + \chi_{S^C}(t_0).([\![f_2]\!]_t\ \mathit{env}) && (6)\\
[\![\mathrm{lfp}(\mathit{name} \mapsto f)]\!]_t.\mathit{env} &= \mathrm{lfp}(\lambda\phi.[\![f]\!]_t.\mathit{env}[\mathit{name} \mapsto \phi]) && (7)\\
[\![\mathrm{gfp}(\mathit{name} \mapsto f)]\!]_t.\mathit{env} &= \mathrm{gfp}(\lambda\phi.[\![f]\!]_t.\mathit{env}[\mathit{name} \mapsto \phi]) && (8)\\
[\![\mathrm{shift}(f)]\!]_t.\mathit{env} &= ([\![f]\!]_t.\mathit{env}) \circ \mathrm{shift} && (9)\\
[\![\text{let } \mathit{name} = f_1 \text{ in } f_2]\!]_t.\mathit{env} &= [\![f_2]\!]_t.\mathit{env}[\mathit{name} \mapsto [\![f_1]\!]_t.\mathit{env}] && (10)
\end{aligned}
$$

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 $\boldsymbol{t} \in \Omega^{\mathbb{N}}$.

Let us note that we could introduce a negation operator $\neg$ ($[\![\neg f]\!]_t = 1 - [\![f]\!]_t$) as syntactic sugar — any formula with negations can be transformed into one without by pushing the negations to the leaves.

Lemma 6 *For all formula f, $[\![f]\!]_t$ is monotone. For all formula f without* lfp *or* gfp, *$[\![f]\!]_t$ is continuous.*

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:

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

– 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:

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

– 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.

– 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 [10, sect. 5]:

$$\mathcal{R} = \bigvee_{i=0}^{k-1} (\Box\Diamond U_{2i} \wedge \neg\Box\Diamond U_{2i+1}) \tag{14}$$

It corresponds to the following formula:

$$\mathrm{gfp}(x_{2k-1} \mapsto \mathrm{lfp}(x_{2k} \mapsto \cdots \mathrm{gfp}(x_1 \mapsto \mathrm{lfp}(x_0 \mapsto ((\cdots(x_{2k-1} + U_{2k-1}x_{2k-2})\cdots +_{U_1} x_0) +_{U_0} 0) \tag{15}$$

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

$$[\![\Sigma A]\!]_t : \left| \begin{array}{ll} \Omega^{\mathbb{N}} & \to [0, +\infty] \\ (x_n)_{n\in\mathbb{N}} & \mapsto \sum_{k=0}^{\infty} f(x_k) \end{array} \right. \tag{16}$$

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

$$[\![\Sigma f]\!]_t = \mathrm{lfp}(\phi \to f + \phi \circ \mathrm{shift}) \tag{17}$$

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").

4.3 Temporal logics

Temporal logics [5, chapter 3] are expressive means of specifying properties of transition systems.

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 [5, §3.2], or, equivalently, by a deterministic Rabin automaton R [25, §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 $E_+^{S\times R}([\![C]\!]_t)$ [10, §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 $E_+^{S\times B}([\![C]\!]_t)$ [10, §4].[2]

Branching-time logic: pCTL and pCTL* The branching-time logics CTL and CTL* [5, §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 [11] 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

$$[\![\mathtt{f}]\!]_s = \{x_0 \in \Omega \mid \exists x_1, \ldots \ \langle x_0, x_1, \ldots\rangle \in [\![\mathtt{f}]\!]_p\}. \tag{18}$$

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:

$$[\![f_{\bowtie\alpha}]\!] = \{x_0 \in \Omega \mid \forall (U_n)_{n\in\mathbb{N}} S([\![f]\!]_t, (U_n)_{n\in\mathbb{N}})(x_0) \bowtie \alpha\} \tag{19}$$

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):

$$\forall x_0 \, [\![f]\!]^\sharp_{e+}(x_0) \bowtie \alpha \implies x_0 \notin [\![f_{\bowtie\alpha}]\!]. \tag{20}$$

[2] Note that this does not hold for nondeterministic Büchi automata, since the automaton is allowed to take its nondeterministic choices with the knowledge of the full sequence of states, not only the past and present states.

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 $E_+(\llbracket f \rrbracket_t)$, or at least get an upper bound for it. Unfortunately, the definitions of both $\llbracket f \rrbracket_t$ and E_+ do not yield effective means to do so.

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

5.1 Backwards Worst Case Semantics on Expectation Functions

A well-known solution to the problem of the optimal value of a Markov decision process is *value iteration* [23, §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 [23, §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.

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.

$\llbracket \mathit{formula} \rrbracket_{e+} : (\Omega \to I) \to (\Omega \to I)$ is defined inductively as follows:

$$\llbracket \mathit{name} \rrbracket_{e+} .\mathit{env} = \mathit{env}(\mathit{name}) \tag{21}$$

$$\llbracket \mathit{constant} \rrbracket_{e+} .\mathit{env} = \lambda x.\mathit{constant} \tag{22}$$

$$\llbracket f_1 +_S f_2 \rrbracket_{e+} .\mathit{env} = \chi_S.(\llbracket f_1 \rrbracket_{e+}\ \mathit{env}) + \chi_{S^C}.(\llbracket f_2 \rrbracket_{e+}\ \mathit{env}) \tag{23}$$

$$\llbracket \mathrm{lfp}(\mathit{name} \mapsto f) \rrbracket_{e+} .\mathit{env} = \mathrm{lfp}(\lambda\phi.\llbracket f \rrbracket_{e+} .\mathit{env}[\mathit{name} \mapsto \phi]) \tag{24}$$

$$\llbracket \mathrm{gfp}(\mathit{name} \mapsto f) \rrbracket_{e+} .\mathit{env} = \mathrm{gfp}(\lambda\phi.\llbracket f \rrbracket_{e+} .\mathit{env}[\mathit{name} \mapsto \phi]) \tag{25}$$

$$\llbracket \mathrm{shift}(f) \rrbracket_{e+} .\mathit{env} = \sup_{T \in \mathcal{T}} (\overleftarrow{T}(\llbracket f \rrbracket_{e+} .\mathit{env})) \tag{26}$$

$$\llbracket \text{let } \mathit{name} = f_1 \text{ in } f_2 \rrbracket_{e+} .\mathit{env} = \llbracket f_2 \rrbracket_{e+} .\mathit{env}[\mathit{name} \mapsto \llbracket f_1 \rrbracket_{e+} .\mathit{env}] \tag{27}$$

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.

$\llbracket \mathit{formula} \rrbracket_{e+}$ is monotone.

Lemma 7 *Let f be a formula not containing* gfp. $\llbracket f \rrbracket_{e+}$ *is ω-upper-continuous.*

As for the summation valuator,

$$\llbracket \Sigma f \rrbracket_{e+} = \mathrm{lfp}\left(\phi \mapsto f + \sup_{T \in \mathcal{T}} (\overleftarrow{T}.\phi)\right) \tag{28}$$

5.2 The Abstraction Relation Between the Semantics

Theorem 8 *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,*

$$[\![f]\!]_{e+}.(E_+(env)) = E_+([\![f]\!]_t.env) \tag{29}$$

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 $t_0 \in X$.

$$\begin{array}{ll}
& [\![f_1 +_S f_2]\!]_{e+}.(E_+(env)).t_0 \\
= & \chi_S(t_0).([\![f_1]\!]_{e+}.E_+(env).t_0) + \chi_{S^C}(t_0).([\![f_2]\!]_{e+}.E_+(env).t_0) \\
= & \chi_S(t_0).(E_+([\![f_1]\!]_t.env).t_0) + \chi_{S^C}(t_0).(E_+([\![f_2]\!]_t.env).t_0) & \text{(induction)} \\
= & E_+(\lambda t.\chi_S(t_0).([\![f_1]\!]_t\, envt) + \chi_{S^C}(t_0).([\![f_2]\!]_t\, envt)).t_0 & \text{(lemma 2)} \\
= & E_+([\![f_1 +_S f_2]\!]_t).t_0.
\end{array}$$

- 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 $\overleftarrow{T_1}.E_+([\![f]\!]_t)$. From lemma 1, $\overleftarrow{T_1}$ is a monotonic, ω-continuous, operator;
  from lemma 3, $R([\![f]\!]_t)$ is directed;
  from lemma 4,

$$\bigsqcup_{f \in R([\![f]\!]_t.env)} (\overleftarrow{T_1} f) = \overleftarrow{T_1}\Big(\bigsqcup_{f \in R([\![f]\!]_t.env)} f \Big).$$

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

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

  Let us apply lemma 11 to that last expression. We obtain

$$\overleftarrow{T_1}.E_+([\![f]\!]_t).t_0 = \sup_{(U_n)_{n\geq 2}} \int \lambda\langle t_2, \ldots\rangle.([\![f]\!]_t.env)(\langle t_1, \ldots\rangle)\, \mathrm{d}[G(T,(U_n)_{n\geq 1}).t_0] \tag{30}$$

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

$$\begin{array}{l}
E_+([\![\mathrm{shift}(f)]\!]_t.env).t_0 \\
= \sup_{(U_n)_{n\geq 1}} \int \lambda\langle t_1, \ldots\rangle.([\![f]\!]_t.env) \circ \mathrm{shift}(\langle t_0, t_1, \ldots\rangle)\, \mathrm{d}[G(T,(U_n)_{n\geq 1}).t_0] \\
= \sup_{U_1} \sup_{(U_n)_{n\geq 2}} \int \lambda\langle t_1, \ldots\rangle.([\![f]\!]_t.env)(\langle t_1, \ldots\rangle)\, \mathrm{d}[G(T,(U_n)_{n\geq 1}).t_0] \\
= \Big(\sup_{U_1} \overleftarrow{\big(T \circ \left[\begin{smallmatrix}\mathrm{Id}\\ U_1\end{smallmatrix}\right]\big)}.E_+([\![f]\!]_t)\Big).t_0 \quad \text{(using Equ. 30)} \\
= [\![\mathrm{shift}(f)]\!]_{e+}.E_+(env)
\end{array}$$

– $[\![\mathrm{lfp}(name \mapsto f)]\!]_{e+}.env = \mathrm{lfp}(\lambda\phi.[\![f]\!]_{e+}.env[name \mapsto \phi])$.
From lemma 7, $\lambda\phi.[\![f]\!]_{e+}.env[name \mapsto \phi]$ is ω-upper-continuous.
$[\![\mathrm{lfp}(name \mapsto f)]\!]_{t}.env = \mathrm{lfp}(\lambda\phi.[\![f]\!]_{t}.env[name \mapsto \phi])$.
From lemma 6, $\lambda\phi.[\![f]\!]_{t}.env[name \mapsto \phi]$ 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 5, E_+ is ω-upper-continuous. The conclusion then follows from lemma 5.

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

Theorem 9 *Let f be a formula. Let env be an environment of valuators. Let us suppose that $H \geq E_+(env)$ pointwise. Then*

$$[\![f]\!]_{e+}.(H) \geq E_+([\![f]\!]_t.env). \tag{31}$$

Proof by induction similar to that of Th. 8. Also similarly we can guarantee the soundness of the analysis of summations:

Theorem 10 *The semantics of the summing operator satisfies:*

$$E_+([\![\Sigma f]\!]_t) = [\![\Sigma f]\!]_{e+}. \tag{32}$$

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:

$$\forall env\ \forall env^\sharp\ env^\sharp \geq env \implies [\![f]\!]^\sharp_{e+}.env^\sharp \geq [\![f]\!]_{e+}.env. \tag{33}$$

The computations for $[\![f]\!]^\sharp_{e+}$ will be done symbolically in an *abstract domain* such as the ones described in [19, 20].

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

$$\forall f,\ \forall T \in \mathcal{T},\ \mathrm{pre}^\sharp.f^\sharp \geq T^*.f^\sharp. \tag{34}$$

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

$$\forall env,\ \forall env^\sharp,\ env^\sharp \geq env \implies [\![\mathrm{shift}(f)]\!]^\sharp_{e+}.env^\sharp \geq [\![\mathrm{shift}(f)]\!]_{e+}.env. \tag{35}$$

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* [9, §4.3]. A widening operator ∇ is a kind of abstraction of the least upper bound that enforces convergence:
  - $f \nabla g \geq \sup(f, g)$ (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* [9, §4.3].
– We shall approximate greatest fixpoints using a limited iteration approach: if $f^\sharp$ is an upper approximation of f, then for any $n \in \mathbb{N}$, ${f^\sharp}^n(\top) \geq \operatorname{gfp} f$.

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. 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 $L^\sharp$.

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:

$$F : \left| \begin{array}{l} (P \times M \to I) \to (P \times M \to I) \\ h \qquad\qquad\quad \mapsto (l,m) \mapsto \sup_{y\in Y} \sum_{(l',m')\in P\times M} T\left((l,m), y; (l',m')\right) . h(l',m') \end{array} \right. \tag{36}$$

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

$$(F.h)(l) = \text{choice}^*_{Y_l} \circ \text{random}^*_{R_l} \circ (F_l)^*_p \left(\sum_{l'\in P} \phi^*_{\tau_l l'} \left(h(l', \bullet) \right) \right) \tag{37}$$

using the following building blocks:

$$\mathrm{choice}^*_{Y_l}(h) = m \mapsto \sup_{y \in Y_l} h(m, y) \tag{38}$$

$$\mathrm{random}^*_{R_l}(h) = m \mapsto \int h(m, r)\, \mathrm{d}\mu_{\mathcal{R}_l}(r) \tag{39}$$

$$(F_l)^*_p(h) = h \circ F_l \tag{40}$$

$$\phi^*_A(h) = h.\chi_A \tag{41}$$

The reasons for those notations are explained in earlier works on the linear adjoint of Kozen's denotational semantics for probabilistic programs [19].

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 the function F is ω-upper-continuous, the least fixpoint of F is obtained as the limit of $F^n(0)$ (let us recall that 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:

$$f_1^{(n+1)} = F_1(f_1^{(n)}, \ldots, f_{|P|}^{(n)}) \tag{42}$$

$$\vdots \tag{43}$$

$$f_{|P|}^{(n+1)} = F_{|P|}(f_1^{(n)}, \ldots, f_{|P|}^{(n)}) \tag{44}$$

$$\tag{45}$$

In terms of implementation, this means that we update in parallel the $|P|$ elements of the vector representing the iterate. As noted by Cousot [8, §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:

$$\begin{array}{ll} f_1^{(n+1)} & = F_1(f_1^{(n)}, \ldots, f_{|P|}^{(n)}) \\ f_2^{(n+1)} & = F_2(f_1^{(n+1)}, \ldots, f_{|P|}^{(n)}) \\ \vdots & \vdots \\ f_{|P|}^{(n+1)} & = F_{|P|}(f_1^{(n+1)}, \ldots, f_{|P|}^{(n)}) \end{array} \tag{46}$$

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 Conclusion, Related Works 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 [23] 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 [14].

Cleaveland's work [6] 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 [18].

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* [23, §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* [23, §7.3], where the reward -1 is granted the first time the process leaves the set of states to consider.

Formal languages similar to the one we consider have been introduced by other authors, such as *quantitative game μ-calculus* [10]. 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 interpretation 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* [23, 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 [12, 13].

References

1. Luca de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, Department of Computer Science, June 1998. CS-TR-98-1601.
2. 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*. Springer, 1997.
3. 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.
4. 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.
5. Edmund M. Clarke, Jr, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, 1999.
6. 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.
7. C. Courcoubetis and M. Yannakakis. Markov decision processes and regular events. In *Proc. ICALP'90*, volume 443 of *LNCS*, pages 336–349. Springer, 1990.
8. 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.
9. Patrick Cousot and Radhia Cousot. Abstract interpretation and application to logic programs. *J. Logic Prog.*, 2-3(13):103–179, 1992.

10. L. de Alfaro and R. Majumdar. Quantitative solution of omega-regular games. In *STOC'01, 33rd Annual ACM Symposium on Theory of Computing*. ACM, 2001.
11. Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reability. Technical Report R90-13, Swedish Institute of Computer Science, December 1990.
12. 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.
13. 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. Springer, 2000.
14. 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.
15. 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.
16. 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.
17. 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.
18. 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.
19. 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.
20. 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.
21. Jacques Neveu. *Mathematical Foundations of the Calculus of Probabilities*. Holden-Day, 1965.
22. Jacques Neveu. *Bases mathématiques du calcul des probabilités*. Masson et Cie, Éditeurs, Paris, 1970. Préface de R. Fortet. Deuxième édition, revue et corrigée.
23. Martin L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. Wiley series in probability and mathematical statistics. John Wiley & Sons, 1994.
24. Roberto Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Massachusetts Institute of Technology, 1995. Technical report MIT/LCS/TR-676.
25. W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, vol. B*, pages 135–191. Elsevier, 1990.

A Technical lemmas

Lemma 11 *For all f, t_0 and U_1,*

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

Lemma 2. *For all t_0 in$X$,*

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

Lemma 3. *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 $g_3(\boldsymbol{t}) = g_1(\boldsymbol{t})$ if $t_0 \in A$, $g_3(\boldsymbol{t}) = g_2(\boldsymbol{t})$ otherwise, belongs to $R(f)$.*

Lemma 4. *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 $\phi(\sqcup K) = \bigsqcup_{f\in K} \phi(f)$.*

Lemma 5. *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 $\alpha(\mathrm{lfp}\,\psi_1) = \mathrm{lfp}\,\psi_2$.*

B Ionescu Tulcea's Theorem

The intuitive meaning of this theorem [21, 22, 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 12 (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 $\left(\prod_{s=0}^t E_s, \bigotimes_{s=0}^t \mathcal{F}_s\right)$ 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:

$$P_{x_0}\left[\prod_t F_t\right] = \chi_{A_0}(x_0) \int_{x_1\in F_1} P_1^0(x_0; \mathrm{d}x_1) \int_{x_2\in F_2} P_2^{0,1}(x_0, x_1; \mathrm{d}x_2) \cdots \int_{x_T\in F_T} P_T^{0,\ldots,T-1}(x_0, \ldots, x_{T-1}; \mathrm{d}x_T) \quad (47)$$

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:

$$\int_{\Omega} Y(\omega') P_{x_0}(\mathrm{d}\omega') = \int_{F_1} P_1^0(x_0; \mathrm{d}x_1) \int_{F_2} P_2^{0,1}(x_0, x_1; \mathrm{d}x_2) \cdot \int_{x_T \in F_T} Y(x_0, \ldots, x_T) P_T^{0,\ldots,T-1}(x_0, \ldots, x_{T-1}; \mathrm{d}x_T) \quad (48)$$

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

$$x_0 \mapsto \int Y(\omega') P_{x_0}(\mathrm{d}\omega') \quad (49)$$

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